



CQML(Contrôle Qualité et Maintenance Logiciel)

Evaluation du code de Cheikh Gueye Wane en DIC1

Membres du groupe:

- Mor SECK
- Benjamin TANGOMO
- Arfang FAYE
- Fasso Mathias NIAMY
- El Moctar MENDAH
- Djibril Diagne

(Master 1 glsi)

Professeur:
Mr. Ngom LAHAD

1) *Facteurs et critères d'évaluation*

L'évaluation de la **qualité** intrinsèque du logiciel est effectuée sur le produit en développement en fonction des facteurs de qualité attendus, définis lors de la commande (spécifications).

Il s'agit ici d'évaluer le **code** de Cheikh Gueye Wane en DIC1 en se basant sur des **facteurs** et **critères** d'évaluations

Un **facteur** est une caractéristique du logiciel, du processus ou du service contribuant à sa qualité telle qu'elle est ressentie par l'utilisateur.

Un **critère** est un attribut du logiciel par l'intermédiaire duquel un facteur peut être obtenu. C'est également une caractéristique du logiciel sur laquelle le développeur peut agir. (par exemple, sa simplicité)



1) Facteurs et critères d'évaluation

Une **métrique** est la mesure d'une propriété d'un critère. (par exemple, la taille d'un module pour le critère "Simplicité").

Les mesures

Numéro de mesure	Code métrique	Numéro phase	Valeur
1	Commentaires	4	1
2	Nombre Si imbriqués	4	2
3	Nom des variables	4	1
4	Nb lignes par module	4	1



1) Facteurs et critères d'évaluation

Les valeurs des métriques sont obtenues de la façon suivante :

Code métrique	Valeurs lues	Tranches	Valeur de la métrique
Commentaires	10/100	$\geq 20\%$ 2 $< 20\%$ et $\geq 10\%$ 1 $< 10\%$ 0	1
Nombre Si imbriqués	3	≤ 3 2 > 3 et < 5 1 > 5 0	2
Nom des variables	Moyens	Incompréhensibles 0 Moyens 1 Significatifs 2	1
Nombre lignes par module	> 50 et < 100	< 50 2 > 50 et < 100 1 > 100 0	1



1) Facteurs et critères d'évaluation

Justifications :

- **commentaires** : 10/100 car toutes les classes ont été commentées et la plupart des méthodes aussi en prenant en compte les valeurs de retour. Certaines variables ont été commentées. Mais les constructeurs n'ont pas été commentés de même que les getters et setters.
- **Nombre de Si imbriqués** : 3. Car le programme compte exactement 3 Si imbriqués.
- **Nom des variables** : Moyen. Car nous avons trouvé que la plupart pluspart des variables étaient assez compréhensibles mais il y avait toujours certaines variables dont le nom ne permettait pas de décrire le contenu.



1) Facteurs et critères d'évaluation

Les critères

L'évaluation des critères pour cette phase s'effectue avec les coefficients suivants :

<i>Nom du critère</i>	<i>Code métrique</i>	<i>Numéro phase</i>	<i>Coefficient</i>
Autodocumentation	Commentaires	4	0.5
Autodocumentation	Nom des variables	4	0.5
Simplicité	Commentaires	4	0.4
Simplicité	Nombre de Si imbriqués	4	0.4
Simplicité	Nombre de lignes d'un module	4	0.2



1) Facteurs et critères d'évaluation

Les facteurs

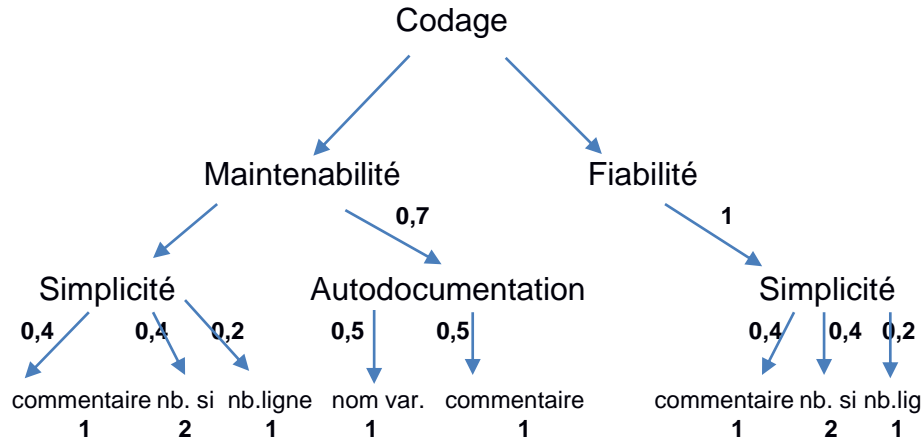
L'évaluation des facteurs pour cette phase s'effectue avec les coefficients suivants :

<i>Nom du facteur</i>	<i>Nom du critère</i>	<i>Numéro phase</i>	<i>Coefficient</i>
Maintenabilité	Simplicité	4	0.3
Maintenabilité	Autodocumentation	4	0.7
Fiabilité	Simplicité	4	1



1) Facteurs et critères d'évaluation

□ Représentation de l'arborescence de la méthode d'évaluation.



1) Facteurs et critères d'évaluation

❑ Calcul de la valeur de chaque **critère** :

Valeur Autodocumentation = $(1 * 0,5) + (1 * 0,5) = 1$

Valeur Simplicité = $(1 * 0,4) + (2 * 0,4) + (1 * 0,2) = 1,4$

❑ Calcul de la valeur de chaque **facteur** :

Valeur Maintenabilité = $(1 * 0,7) + (1,4 * 0,3) = 1,12$

Valeur Fiabilité = $(1,4 * 1) = 1,4$



1) Facteurs et critères d'évaluation

- ❑ Calcul de la valeur de la **qualité totale** (qt) du Code de Cheikh Gueye Wane pour la phase 4 :

$$qt = (1,12 * 0,5) + (1,4 * 0,5) = 1,26$$

- ❑ Valeur **maximale** possible (maxqt) :

$$\text{Valeur Autodocumentation} = (2 * 0,5) + (2 * 0,5) = 2$$

$$\text{Valeur Simplicité} = (2 * 0,4) + (2 * 0,4) + (2 * 0,2) = 2$$

$$\text{Valeur Maintenabilité} = (2 * 0,7) + (2 * 0,3) = 2$$

$$\text{Valeur Fiabilité} = (2 * 1) = 2$$

$$\text{maxqt} = (2 * 0,5) + (2 * 0,5) = 2$$



2) Programme de test Unitaire avec JAVA

Le **test Unitaire** est une procédure permettant de vérifier le bon fonctionnement d'une partie précise d'un logiciel ou d'une portion d'un programme.

Il s'agit donc ici d'évaluer le **code** de Cheikh Gueye Wane sous le nom de **2nddegre** qui est un programme qui a été réalisé en JAVA (la bibliothèque **javafx**) qui permet de résoudre une équation de second degré. Ce programme comporte les fonctionnalités suivantes :

- Le bouton **résoudre** qui permet de résoudre une équation de second degré.
- les **champs de texte** destinés à contenir les valeurs et le résultat.



2) Programme de test Unitaire avec JAVA

- le bouton **FR/ENG** pour changer la langue
- le bouton **Effacer** pour réinitialiser les champs de texte.
- le bouton **Menu/historique** pour afficher l'historique des résolutions effectuées.
- le bouton **Menu/help** pour afficher l'aide.

Pour effectuer ces test Unitaire, nous avons eu à importer plusieurs bibliothèques : **Junit**, **assertj-core**, **testfx**, **guava**.



2) Programme de test Unitaire avec JAVA

```
@Test
public void testTranslate() throws Exception{

    System.out.println("**** testTranslate ****");
    JSONObject c = (JSONObject) api.sendGet("get_case/3");
    String step="";
    //verifie si l'objet n'est pas nul.
    assertNotNull(lngButtonId);
    verifyThat(lngButtonId, hasText("FR"));
    clickOn(lngButtonId);
    step+="Cliquer FR 1ere fois\n";
    sleep(1000);
    verifyThat(lngButtonId, hasText("ENG"));
    clickOn(lngButtonId);
    step+= "Cliquer EN 1ere fois\n";
    sleep(1000);
    verifyThat(lngButtonId, hasText("FR"));
    clickOn(lngButtonId);
    step+= "Cliquer FR 2e fois\n";
    sleep(1000);
    verifyThat(lngButtonId, hasText("ENG"));
    clickOn(lngButtonId);
    step+= "Cliquer EN 2e fois";
    sleep(1000);
    verifyThat(lngButtonId, hasText("FR"));

    c.put("custom_steps",step);
    c.put("custom_preconds","Ce test s'assure que les deux langues possibles sont FR/EN");

    Map cc = new HashMap();

    cc.put("status_id","1");
    cc.put("custom_step_results","pass");

    api.sendPost("update_case/3",c);
    api.sendPost("add_result_for_case/3/3",cc);

    System.out.println(c);
}
```


testTranslate sur Eclipse



✓ Méthode **testTranslate**

✓ Exécution de la méthode **testTranslate**



 testTranslate (5,242 s)

502
503
504

C
S
C



2) Programme de test Unitaire avec JAVA

testTranslate

Test Cases

Type	Priority	Estimate
Other	Medium	None
Automation Type		
None		

Preconditions

Ce test s'assure que les deux langues possibles sont FR/EN

Steps

Cliquer FR 1ere fois
Cliquer EN 1ere fois
Cliquer FR 2e fois
Cliquer EN 2e fois

testTranslate sur TestRail



✓ Cas de test **testTranslate**

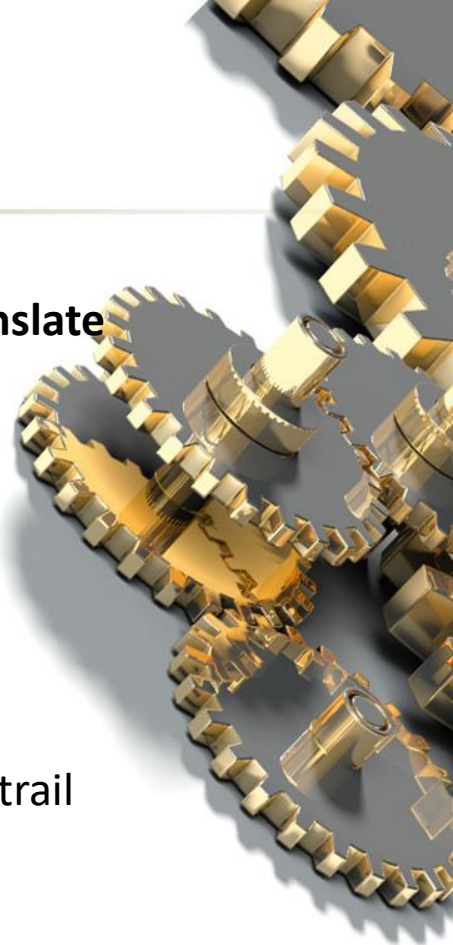
✓ Run **testTranslate** dans Testrail



T5

testTranslate

Passed



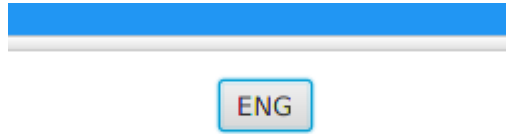
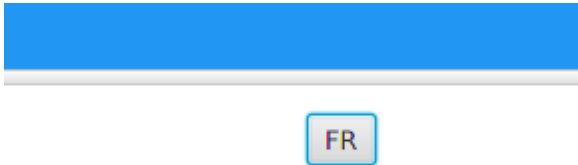
2) Programme de test Unitaire avec JAVA

testTranslate :explication

En effet, le test de cette méthode permet de s'assurer que les deux langues possibles sont **FR/ENG**. C'est à dire si l'utilisateur clique sur le bouton **FR**, il se change en **ENG** et vice versa.

❑ Résultat:

On constate que ce test s'effectué avec succès. Obtient le résultat attendu.



2) Programme de test Unitaire avec JAVA

testTranslateButton sur Eclipse

```
@Test
public void testTranslateButton() throws Exception{
    System.out.println("**** testTranslateButton ****");

    //creation object c
    JSONObject c = (JSONObject) api.sendGet("get_case/4");

    //creation step step
    String step="";

    Button clearButtonId = find("Effacer");

    // verifie que les objets ne sont pas null.
    assertNotNull(lngButtonId);
    assertNotNull(inputLabelId);
    assertNotNull(resultLabelId);
    assertNotNull(submitButtonId);
    assertNotNull(clearButtonId);

    verifyThat(lngButtonId, hasText("FR"));
    step+="Effacer et tester les bouton de FR/EN";
    sleep(2000);
    clickOn(lngButtonId);
    sleep(2000);
    verifyThat(lngButtonId, hasText("ENG"));
    // Verifie si la traduction a bien ete effectuee en ENG.
    verifyThat(inputLabelId, hasText("Enter the value of a,b and c :"));
    verifyThat(resultLabelId, hasText("Result"));
    verifyThat(submitButtonId, hasText("Resoudre"));
    verifyThat(clearButtonId, hasText("Clear"));


    clickOn(lngButtonId);
    sleep(2000);
    verifyThat(lngButtonId, hasText("FR"));
    // Verifie si la traduction a bien ete effectuee en FR.
    verifyThat(inputLabelId, hasText("Entrer les valeurs de a,b et c"));
    verifyThat(resultLabelId, hasText("Resultat"));
    verifyThat(submitButtonId, hasText("Resoudre"));
    verifyThat(clearButtonId, hasText("Effacer"));
```



✓ Méthode **testTranslateButton**

✓ Exécution de la méthode
testTranslateButton



 testTranslateButton (6,946 s)



2) Programme de test Unitaire avec JAVA

testTranslateButton sur TestRail

C4 testTranslateButton

Test Cases

Type Other	Priority Medium	Estimate None
Automation Type None		

Preconditions

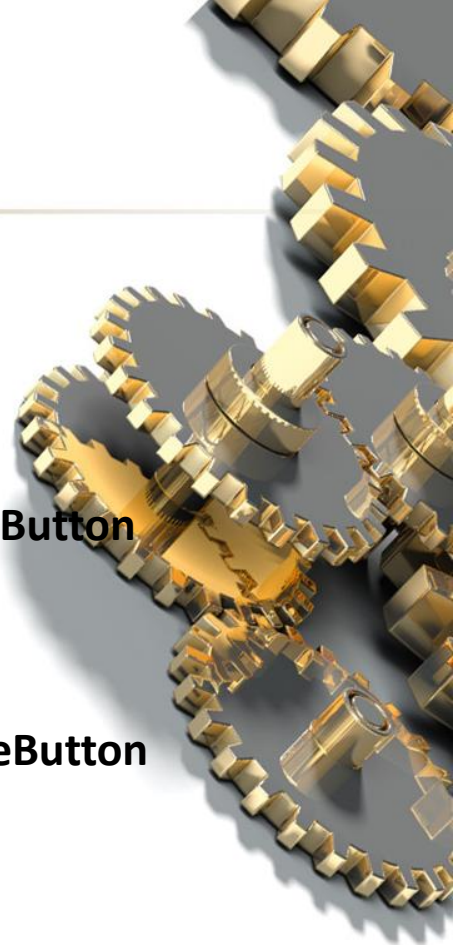
Ce test s'assure que les deux langues possibles sont FR/EN

Steps

Effecer et tester les bouton de FR/EN

✓ Cas de test **testTranslateButton**

✓ Run **testTranslateButton**



2) Programme de test Unitaire avec JAVA

testTranslateButton :explication

Ce test s'assure que la traduction en **FR/ENG** est bien effectuée après un click sur le bouton **FR/ENG**.

Lorsque l'utilisateur clique sur bouton **FR**, on doit avoir la traduction en français et lorsqu'il clique sur **ENG** on doit avoir la traduction en anglais.

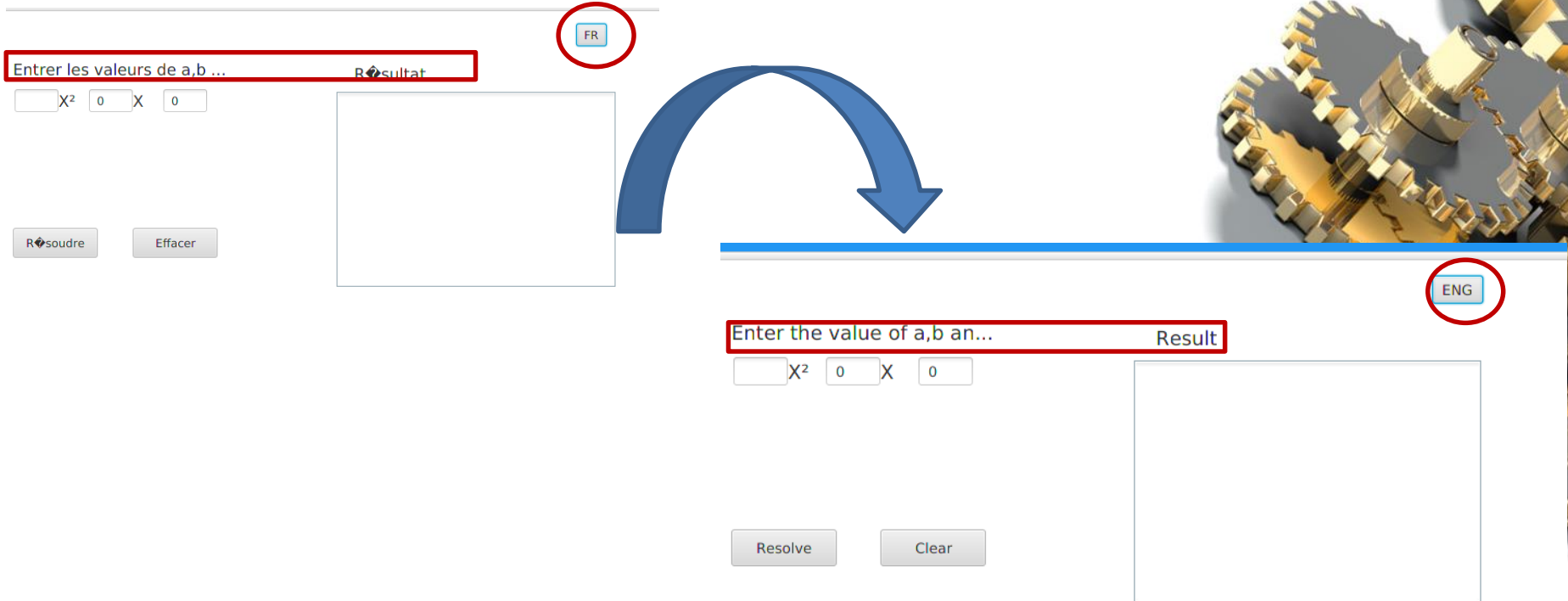
❑ Résultat :

On Constate que ce test S'est effectué avec succès
Le changement de langue est bien pris en compte.



2) Programme de test Unitaire avec JAVA

testTranslateButon :illustration



2) Programme de test Unitaire avec JAVA

testHistoriqueClick sur eclipse

```
@Test
public void testHistoriqueClick() throws Exception{

    //creer object c
    JSONObject c = (JSONObject) api.sendGet("get_case/5");

    //creation step
    String step="";

    System.out.println("**** testHistoriqueClick ****");
    // verifie si les objets ne sont pas nuls.
    assertNotNull(menuButtonId);
    assertNotNull(menuItemId);
    // click sur le bouton 'Menu'.
    clickOn(menuButtonId);
    sleep(2000);
    // click sur le bouton 'historique'.
    clickOn(menuItemId);

    step+="Clique sur bouton historique";

    sleep(2000);

    c.put("custom_steps", step);
    c.put("custom_preconds", "Ce test s'assure qu'une nouvelle fenetre est affichee apres click sur le bouton historique");
    api.sendPost("update_case/5",c);

    //ajouter resultat cas de test
    Map cc = new HashMap();
    cc.put("status_id", "1");
    cc.put("custom_step_results", "pass");
    api.sendPost("add_result_for_case/3/5",cc);
}
```

✓ Méthode **testHistoriqueClick**



2) Programme de test Unitaire avec JAVA

testHistoriqueClick : explication

Lorsque l'utilisateur click que sur le bouton **menu** puis sur le menuitem **historique** on doit avoir l'historiques des valeurs entrées lors de la résolution d'une équation du second degre.

❑ Résultat :

En fait l'interface n'a pas pus s'afficher par ce qu'il n'y avait Aucune donnée. On peut remarquer les données n'ont pas été enregistrées dans la base de données car la table **resolution** est n'a pas été crée. Donc on essaie de récupérer les données sur une table qui n'existe pas.



2) Programme de test Unitaire avec JAVA

testHistoriqueClick : illustration

Application de resolution d'equation 2nd degré dans R

Menu

Historique

Aide

ENG

Enter the value of a,b an...

X² 0 X 0

Resolve Clear

Result



2) Programme de test Unitaire avec JAVA

testhandleHelpClick sur eclipse

```
//id = 6
@Test
public void TesthandleHelpClick() throws Exception{

    //créer object c
    JSONObject c = (JSONObject) api.sendGet("get_case/6");

    //creation step step
    String step="";

    System.out.println("*** testHandleHelpClick ***");
    // vérifie si les objets ne sont pas nuls.
    assertNotNull(menuButtonId);
    assertNotNull(menuHelpItemId);
}
```

← ✓ Méthode **testHandlerClick**

✓ Exécution de la méthode
testHandlerClick



TesthandleHelpClick (17,844 s)

49 /
498



2) Programme de test Unitaire avec JAVA

testhandleHelpClick sur TestRail

✓ TestHandlerClick dans TestRail

C6 TesthandleHelpClick

Test Cases

Type	Other	Priority	Medium	Estimate	None
Automation Type	None				

Preconditions

Ce test s'assure qu'une fenetre d'aide est ouverte apres click sur le bouton help

Steps

Clique sur bouton d'aide ou help



✓ Test **testHandlerClick**

✓ Run **testHandlerClick**



2) Programme de test Unitaire avec JAVA

testhandleHelpClick :explication

Ce test s'assure qu'une fenêtre d'aide est ouverte après click sur le menu puis sur le menuitem **help**.

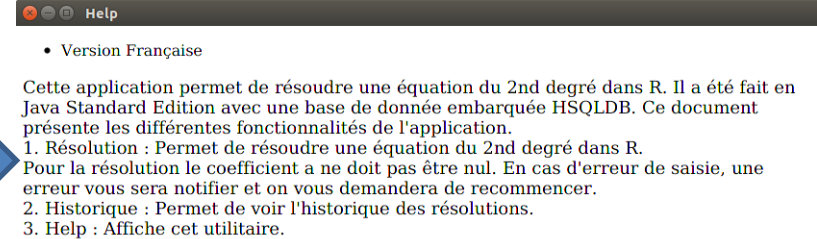
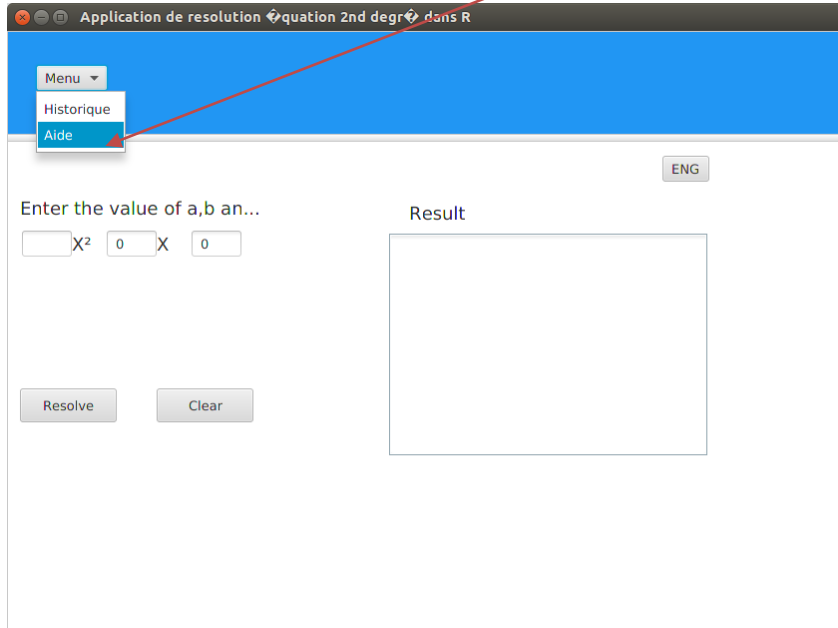
☐ Résultat :

On constate qu'on a le résultat souhaité car après click sur le menuitem Help une fenêtre d'aide s'ouvre et on obtient l'aide voulue.



2) Programme de test Unitaire avec JAVA

testhandleHelpClick : illustration



2) Programme de test Unitaire avec JAVA

testClearButton sur eclipse

```
//id = 7
@Test
public void testClearButton() throws Exception{
```

```
    //creer object c
    JSONObject c = (JSONObject) api.sendGet("get_case/7");
```

```
    //creaton step step
    String step="";
```

```
    System.out.println("*** testClearButton ***");
    TextField firstOpId = find("#firstOp");
    TextField secondOpId = find("#secondOp");
    TextField thirdOpId = find("#thirdOp");
    TextArea resultFieldId = find("#resultField");
```

← ✓ Méthode **testClearButton**

✓ Exécution de la méthode
testClearButton



testClearButton (7,811 s)

501



2) Programme de test Unitaire avec JAVA

testClearButton sur TestRail

T14 testClearButton ▾

Test Run 06/08/2018 > Test Cases

Type	Priority	Es
Other	Medium	No
Automation Type		
None		

Preconditions

Ce test s'assure que les differents champs sont initialises apres click sur le bouton clear

Steps

Clique sur bouton effacer

Results & Comments ? History & Context ? Defects ?

Add a comment ..

Passed

06/08/2018 22:24 Mor S. [Edit](#)

This test was marked as 'Passed'.

T14 testClearButton



✓ Test **testClearButton**

✓ Run **testClearButton**



Passed ▾

2) Programme de test Unitaire avec JAVA

testClearButton : explication

Ce test s'assure que les différents champs sont initialisés après click sur le bouton **clear**. c'est à dire si l'utilisateur click sur le bouton clear et qu'au préalable il y avait les valeurs entrée dans les champs. Ces champs seront **réinitialisés**.

❑ Résultat:

On obtient donc le résultat voulu car après click sur le bouton clear les champs qui contenait des valeurs ont été initialisés.



2) Programme de test Unitaire avec JAVA

testClearButton :illustration

Enter the value of a,b an...

<input type="text"/>	X ²	<input type="text"/>	X	<input type="text"/>
----------------------	----------------	----------------------	---	----------------------

Result



2) Programme de test Unitaire avec JAVA

testResolution sur eclipse

```
@Test
public void testResolution() throws Exception{

    //creer object c
    JSONObject c = (JSONObject) api.sendGet("get_case/8");

    //creaton step step
    String step="";

    System.out.println("*** testResolution ***");
    // Récupération des références vers des id.
    TextField firstOpId = find("#firstOp");
    TextField secondOpId = find("#secondOp");
    TextField thirdOpId = find("#thirdOp");
    TextArea resultFieldId = find("#resultField");
    Button lngButtonId = find("#lngButton");
    Button clearButton = find("Effacer");

    step+="Etap1: recuperation des id\n";


    // vérifie si les objets ne sont pas nuls.
```



✓ Méthode **testResolution**

✓ Exécution de la méthode
testResolution



 testResolution (21,584 s)



2) Programme de test Unitaire avec JAVA

testResolution sur TestRail

T16 testResolution ↕

Test Run 06/08/2018 > Test Cases

Type	Priority	Estimate
Other	Medium	None

Automation Type
None

Preconditions

Ce test s'assure que la methode de resolution resoud sans probleme une equation du second degres

Steps

Etape1: recuperation des id
Etape2: cas ou le systeme admet deux solutions x1 et x2 dans R
Etape3: Cas ou le systeme n'admet pas de solution dans R
Etape4: cas ou le systeme admet une solution double x0 dans R

Results & Comments ⓘ History & Context ⓘ Defects ⓘ

Add a comment ..

T16 testResolution



✓ Test **testResolution**



Run **testResolution**



Passed ▼



2) Programme de test Unitaire avec JAVA

testResolution : Explication

Ce test s'assure que la méthode de résolution résoud sans problème une équation du second degré. En effet l'on doit les avoir les solutions exactes des valeurs entrées ou un message indiquant qu'il n'y a pas de solution pour des valeurs n'ayant pas de solution.

❑ Résultat:

On observe qu'on a bien le résultant attendu.

- ✓ Nous avons utilisé comme valeurs:
2 ; 5 ; 3 et avons trouvé comme solution double:
Les solutions sont $x_1: -1.5$ $x_2: -1.0$ $S = \{-1.5, -1.0\}$
- ✓ puis nous avons testés avec les valeurs: 1; 2 ; 3 ;
dont le système n'admet pas de solution.



2) Programme de test Unitaire avec JAVA

testResolution : explication

- ✓ Enfin avec les valeurs: 4; -12; 9
et comme solution: Le syst?me admet une solution double
x0 :1.5 S = {1.5}

Ors, nous avons pu déceler 2 erreurs à savoir:

- ❖ au niveau de l'affichage, qui affiche la solution avec des caractères non reconnu par ce système d'exploitation.
- ❖ de plus les données ne sont pas enregistrés dans la base de données car la tabe resolution n'existe pas.

user lacks privilege or object not found: RESOLUTION



2) Programme de test Unitaire avec JAVA

testResolution : illustration

Menu ▾

FR

Entrer les valeurs de a,b ...

x^2 x

Résoudre Effacer

Résultat

Les solutions sont x1: -1.5 x2: -1.0



2) Programme de test Unitaire avec JAVA

testResolutionSaisieLettres sur eclipse

```
@Test
public void testResolutionSaisieLettres() throws Exception {

    //creer object c
    JSONObject c = (JSONObject) api.sendGet("get_case/8");

    //creation step step
    String step="";

    // Récupération des références vers des id.
    TextField firstOpId = find("#firstOp");
    TextField secondOpId = find("#secondOp");
    TextField thirdOpId = find("#thirdOp");
    Button clearButton = find("Effacer");
    TextArea resultFieldId = find("#resultField");
```

← ✓ Méthode
testResolutionSaisieLettres

✓ Exécution de la méthode
testResolutionSaisieLettres



testResolutionSaisieLettres (11,208 s)
testResolution (71 504 s)



2) Programme de test Unitaire avec JAVA

testResolutionSaisieLettres sur TestRail

T17 testResolutionSaisieLettres ↕

Test Run 06/08/2018 > Test Cases

Type	Priority
Other	Medium
Automation Type	
None	

Preconditions

Ce test s'assure que la méthode de résolution affiche un message d'erreur lorsqu'e

Steps

Etape1: Recuperation des idEtape2: Verifier les objet non nulsEtape3: Saisi de val

Results & Comments ? History & Context ? Defects ?

Add a comment ..

Passed This test was marked as 'Passed'.

07/08/2018 11:44 Mor S. [Edit](#)

← ✓ Test
testResolutionSaisieLettres

✓ Run **testResolutionSaisieLettres**



T17 testResolutionSaisieLettres

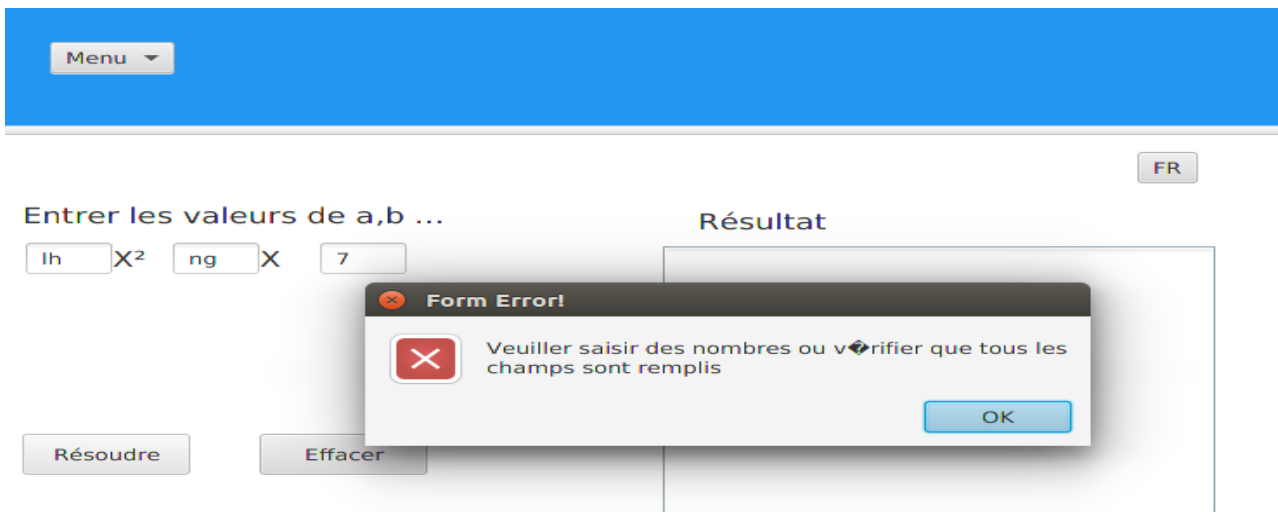
Passed ▾



2) Programme de test Unitaire avec JAVA

testResolutionSaisieLettres : Explication - Illustration

Ce test permet de s'assurer qu'un message d'erreur est affiché à l'utilisateur lorsqu'il saisit des lettres et qu'il valide.



The screenshot displays a web application interface. At the top, there is a blue header bar with a "Menu" dropdown button. Below the header, on the right side, is a language selector button labeled "FR". The main content area is divided into two sections: "Entrer les valeurs de a,b ..." and "Résultat". In the "Entrer les valeurs de a,b ..." section, there are three input fields. The first field contains "lh", the second contains "X²", and the third contains "7". Below these fields are two buttons: "Résoudre" and "Effacer". A modal dialog box titled "Form Error!" is overlaid on the interface. It features a red square icon with a white 'X' and the text "Veuillez saisir des nombres ou vérifier que tous les champs sont remplis". An "OK" button is located at the bottom right of the dialog box. On the right side of the slide, there is a decorative graphic of several interlocking golden gears.

3) Synchronisation du résultat de ce test dans la base de données Mysql avec testrail

✓ Fichier(classe) **APIClient**

```
testrailApi
├── APIClient.java
├── APIException.java
└── IDE Custom Libraries \JDKCE 1.01
```

✓ Code **APIClient**

```
package testrailApi;

import java.net.URL;

public class APIClient
{
    private String m_user;
    private String m_password ;
    private String m_url;

    public APIClient(String base_url)
    {
        if (!base_url.endsWith("/"))
        {
            base_url += "/";
        }
    }
}
```

✓ Mise à jour cas de test – Validation test

```
//update cas de test
c.put("custom_steps", step);
c.put("custom_preconds", "Ce test s'assure que la méthode de résolution affiche un message d'erreur lors de la résolution");
api.sendPost("update_case/9",c);

//ajouter resultat cas de test
Map cc = new HashMap();
cc.put("status_id", "1");
cc.put("custom_step_results", "pass");
api.sendPost("add_result_for_case/9",cc);
```

✓ Instance de l'**APIClient**

```
*/
public class TestFx extends ApplicationTest {

    //on instancie l'api
    public APIClient api = new APIClient("http://localhost/CQML/testrail");

    /**
     * *****
     * Se charge de lancer l'application avant le début des tests.
     * @throws Exception
     * *****
     */
    @Before
    public void setUpClass() throws Exception{
        api.setUser("mor0093@gmail.com");
        api.setPassword("toubu");
        ApplicationTest.launch(Main.class);
    }
}
```

✓ Recupération d'un cas de test

```
//creer object c
JSONObject c = (JSONObject) api.sendGet("get_case/8");
```

4) Choix d'un serveur git (Gestion de configuration) (ici avec capture à l'appui)

Nous faisons le choix de **github** comme serveur git.

Pourquoi:

- ✓ Plus facile à administrer
- ✓ Rapide à mettre en place
- ✓ Faire avancer la communauté open-source

✓ Création de repository

Overview Repositories 2 Stars 0 Followers 0 Following 0

Search repositories...

Type: All ▼

Language: All ▼



Create a new repository

A repository contains all the files for your project, including the revision history.

Owner



morseck ▼

Repository name

projet_final_testRail ✓

Great repository names are short and memorable. Need inspiration? How about [symmetrical-palm-tree](#).

✓ Lien du repository sur Github

Quick setup — if you've done this kind of thing before

Set up in Desktop

or

HTTPS

SSH

https://github.com/morseck/projet_final_testRail.git

We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

4) Choix d'un serveur git (Gestion de configuration) (ici avec capture à l'appui)

- ✓ Initialiser le repertoire comme projet git

```
E:\COURS\MASTER 1\SEMESTRE2\CQML\Resolution_equation_2nd_Degres>git init  
Initialized empty Git repository in E:/COURS/MASTER 1/SEMESTRE2/CQML/Resolution_equation_2nd_Degres/.git/
```

- ✓ Traquer les fichier

```
E:\COURS\MASTER 1\SEMESTRE2\CQML\Resolution_equation_2nd_Degres>git add -A  
warning: LF will be replaced by CRLF in 2ndDegres_Programme/TestSecondDegre.java.  
The file will have its original line endings in your working directory.
```

- ✓ Faire un commit du projet

```
E:\COURS\MASTER 1\SEMESTRE2\CQML\Resolution_equation_2nd_Degres>git commit -m "First Commit"  
[master (root-commit) acd6651] First Commit  
106 files changed, 41644 insertions(+)
```

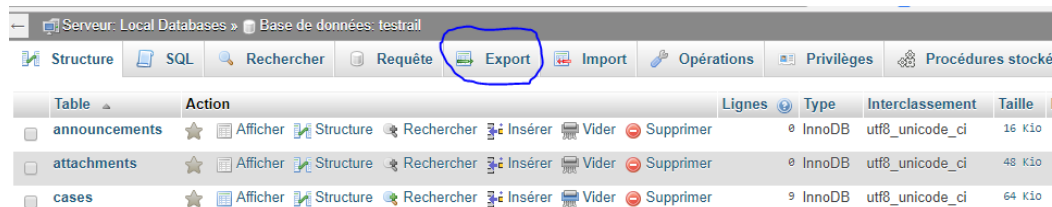
- ✓ Relier le projet avec le repos git sur Github

```
E:\COURS\MASTER 1\SEMESTRE2\CQML\Resolution_equation_2nd_Degres>git remote add origin https://github.com/morseck/projet_final_testRail.git
```



5) *Faites un push de votre programme de test, l'importation de la base de données testrail et du rapport de test. (ici avec capture à l'appui)*

✓ Exportation base de données



Exportation des tables depuis la base de données «testrail»

Méthode d'exportation :

- ☒ Rapide - n'afficher qu'un minimum d'options
- ☐ Personnalisée - afficher toutes les options possibles

Format :

SQL

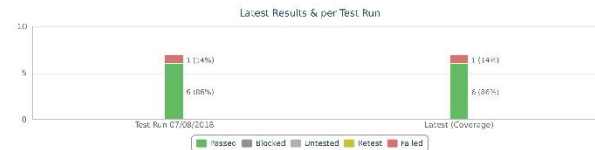
Exécuter

✓ Rapport Test sur TestRail

rapport second degres

Page(s): second Degré
By: sergeyev.sergeyev@... 07/06/2018 14:08

Ce rapport contient les résultats des tests effectués.



Test Runs

The statistics and percent numbers of this report only include the test cases that match the selected filters, if any.

mon 7 août 2018

Test Run 07/06/2018

86%

	Test Run 07/06/2018	Latest (Coverage)
Total	7	7
Passed	6 (86%)	6 (86%)
Skipped	0 (0%)	0 (0%)
Untested	0 (0%)	0 (0%)
Retest	0 (0%)	0 (0%)
Failed	1 (14%)	1 (14%)

Comparison & Coverage

1. Test Cases (7)

ID	Title	Test Run 07/06/2018	Latest (Coverage)
C14	testTranslance	Passed	Passed
C15	testTranslance	Passed	Passed
C16	testTranslance	Failed	Failed
C17	testTranslance	Passed	Passed
C18	testTranslance	Passed	Passed
C19	testTranslance	Passed	Passed
C20	testTranslance	Passed	Passed

Generated with TestRail test management software - 3.5.0.1735
Report: Comparison for Cases (Reports), by Gurock Software (Version 1)

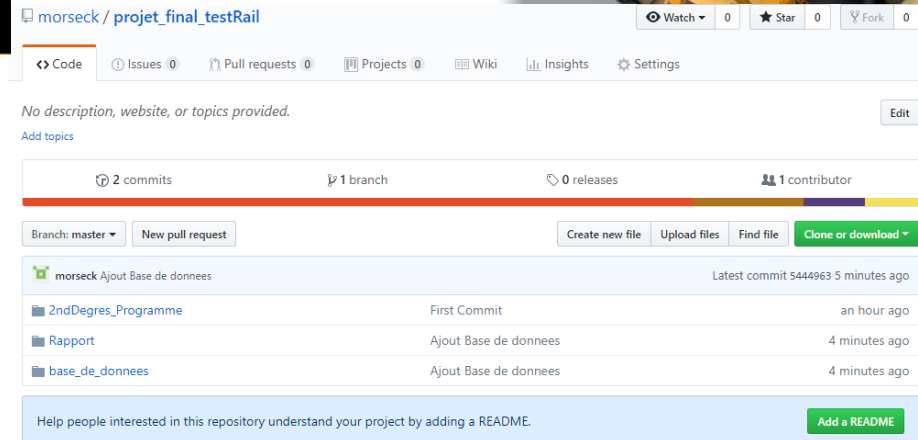
5) *Faites un push de votre programme de test, l'importation de la base de données testrail et du rapport de test. (ici avec capture à l'appui)*

✓ Push du projet

```
E:\COURS\MASTER 1\SEMESTRE2\CQML\Resolution_equation_2nd_Degres>git push -u origin master
fatal: HttpRequestException encountered.
Une erreur s'est produite lors de l'envoi de la demande.
Username for 'https://github.com': mor0093@gmail.com
Password for 'https://mor0093@gmail.com@github.com':
Counting objects: 129, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (120/120), done.
Writing objects: 100% (129/129), 1.69 MiB | 925.00 KiB/s, done.
Total 129 (delta 34), reused 0 (delta 0)
remote: Resolving deltas: 100% (34/34), done.
To https://github.com/morseck/projet_final_testRail.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
```

```
E:\COURS\MASTER 1\SEMESTRE2\CQML\Resolution equation 2nd Degres>
```

✓ Repository sur Github



The screenshot shows the GitHub interface for the repository 'morseck/projet_final_testRail'. At the top, there are navigation tabs for 'Code', 'Issues', 'Pull requests', 'Projects', 'Wiki', 'Insights', and 'Settings'. Below these, a message states 'No description, website, or topics provided.' with an 'Add topics' link and an 'Edit' button. A summary bar indicates '2 commits', '1 branch', '0 releases', and '1 contributor'. Below this, there are buttons for 'Branch: master', 'New pull request', 'Create new file', 'Upload files', 'Find file', and 'Clone or download'. The commit history table shows three entries: 'morseck Ajout Base de donnees' (Latest commit 5444963 5 minutes ago), '2ndDegres_Programme' (First Commit an hour ago), 'Rapport' (Ajout Base de donnees 4 minutes ago), and 'base_de_donnees' (Ajout Base de donnees 4 minutes ago). At the bottom, there is a prompt to 'Help people interested in this repository understand your project by adding a README.' with an 'Add a README' button.

Commit	Message	Time
morseck	Ajout Base de donnees	Latest commit 5444963 5 minutes ago
	2ndDegres_Programme	First Commit an hour ago
	Rapport	Ajout Base de donnees 4 minutes ago
	base_de_donnees	Ajout Base de donnees 4 minutes ago

6) Recommandations pour améliorer la qualité du code.

Nous avons jugé nécessaire de présenter quelques recommandations pour améliorer le code de Cheikh Gueye Wane en DIC1 à savoir :

- ✓ Utiliser plus de commentaires, commentés aussi les variables car ce sont ces commentaires qui vont permettre de décrire le contenu de la variable.
- ✓ De même éviter aussi trop les imbrications de condition si
- ✓ Utiliser des noms de variables significatifs
- ✓ Utiliser un encodage de caractère qui permettra de représenter n'importe quel caractère saisi.



FIN !!

