AUDIT MYERP

OpenClassRooms P9

Frédéric Leroux

TABLE DES MATIÈRES

AIM

This document presents all the elements and actions put in place in order to:

- Proceed to the project audit
- Put in place a continuous integration and delivery
- Put in place a testing strategy
- Put in place coverage survey
- Implement TODO tasks
- Correct eventual issues, mistakes, bugs

In the project Project_B4 MYERP ongoing development context.
Moreover, this document attempt to put in place a testing and verification methodology.

INTRODUCTION

The project Project_B4 MYERP is a billing and accounting system.
This project audit concerns an ongoing development, and have as objective to check the respect of 7 RG_Compta (accounting management rules) based on the Double-entry bookkeeping,
as well as the well-functioning of the current implemented functionalities with bugs fix.
Plus, the TODO tasks implementation, and finally furnish a global test coverage greater than or equal to 75%.
All the objectives will be traced and made available with a continuous integration and delivery server.

PROGRAMING TOOLS

IDE

USED

Eclipse IDE for Enterprise Java Developers

DEFINITION

See : https://www.eclipse.org/ide/

VERSION

2020-09 (4.17.0)

PROGRAMMING LANGUAGE

APPLIED

object-oriented programming language Java

DEFINITION

AS per: https://www.java.com/en/download/help/whatis_java.html
Java is a programming language and computing platform first released by Sun Microsystems in 1995. There are lots of applications and websites that will not work unless you have Java installed, and more are created every day. Java is fast, secure, and reliable. From laptops to datacenters, game consoles to scientific supercomputers, cell phones to the Internet, Java is everywhere!

VERSION

1.8

FRAMEWORK

APPLIED

SpringFramework

DEFINITION

As per : https://spring.io/projects/spring-framework
The Spring Framework provides a comprehensive programming and configuration model for modern Java-based enterprise applications - on any kind of deployment platform.
A key element of Spring is infrastructural support at the application level: Spring focuses on the "plumbing" of enterprise applications so that teams can focus on application-level business logic, without unnecessary ties to specific deployment environments.

VERSION

5.2.13.RELEASE

BUILD AUTOMATION TOOL

APPLIED

Apache Maven

DEFINITION

As per: https://maven.apache.org/
Apache Maven is a software project management and comprehension tool. Based on the concept of a project object model (POM), Maven can manage a project's build, reporting and documentation from a central piece of information.

VERSION

3.6.0

CONTINUOUS INTEGRATION AND DELIVERY

DEFINITION

In software engineering, continuous integration (CI) is the practice of merging all developers' working copies to a shared mainline several times a day. Grady Booch first proposed the term CI in his 1991 method, although he did not advocate integrating several times a day. Extreme programming (XP) adopted the concept of CI and did advocate integrating more than once per day – perhaps as many as tens of times per day.
As per https://en.wikipedia.org/wiki/Continuous_integration

PROVIDER

For this project we will use Gitlab service as Continuous integration and delivery.
This service provide pipeline based on job, and several tools as:

- coverage tracking
- test report
- bages
- ….

PROJECT REPOSITORY

Clone with HTTPS:
https://gitlab.com/FredLeroux/P9_FLE.git
or SSH
git@gitlab.com:FredLeroux/P9_FLE.git

TEST COVERAGE ANALYZER

DEFINITION

As per: https://www.guru99.com/test-coverage-in-software-testing.html,
https://en.wikipedia.org/wiki/Code_coverage
Test coverage is defined as a metric in Software Testing that measures the amount of testing performed by a set of tests. It will include gathering information about which parts of a program are executed when running the test suite to determine which branches of conditional statements have been taken.
In simple terms, it is a technique to ensure that your tests are testing your code or how much of your code you exercised by running the test.

JACOCO

On this project we will use jacoco plugin, which will provide:

- report
- coverage percent check
- build failure
- site

DEFINITON

As per: https://www.eclemma.org/index.html
EclEmma is a free Java code coverage tool for Eclipse, available under the Eclipse Public License. It brings code coverage analysis directly into the Eclipse workbench:
Fast develop/test cycle: Launches from within the workbench like JUnit test runs can directly be analyzed for code coverage.
Rich coverage analysis: Coverage results are immediately summarized and highlighted in the Java source code editors.
Non-invasive: EclEmma does not require modifying your projects or performing any other setup.
Since version 2.0 EclEmma is based on the JaCoCo code coverage library. The Eclipse integration has its focus on supporting the individual developer in an highly interactive way. For automated builds please refer to JaCoCo documentation for integrations with other tools.

POM PLUGIN

```
<plugin>
        <groupId>org.jacoco</groupId>
        <artifactId>jacoco-maven-plugin</artifactId>
        <version>${jacoco.version}</version>
</plugin>
```

TEST STRATEGIES

UNIT TEST

DEFINITION

As per: https://en.wikipedia.org/wiki/Unit_testing ,
https://www.artofunittesting.com/definition-of-a-unit-test,
https://softwaretestingfundamentals.com/

Unit tests are typically automated tests written to ensure that a section of an application (known as the "unit") meets its design and behaves as intended. In object-oriented programming, a unit is often an entire interface, such as a class, but could be an individual method. By writing tests first for the smallest testable units, then the compound behaviors between those, one can build up comprehensive tests for complex applications.
A good unit test is:

- Able to be fully automated
- Has full control over all the pieces running (Use mocks or stubs to achieve this isolation when needed)
- Can be run in any order if part of many other tests
- Runs in memory (no DB or File access, for example)
- Consistently returns the same result (You always run the same test, so no random numbers, for example. save those for integration or range tests)
- Runs fast
- Tests a single logical concept in the system
- Readable
- Maintainable
- Trustworthy (when you see its result, you don’t need to debug the code just to be sure)

INTEGRATION TEST

DEFINITION

As per: https://en.wikipedia.org/wiki/Integration_testing ,
https://softwaretestingfundamentals.com/

Integration testing (sometimes called integration and testing, abbreviated I&T) is the phase in software testing in which individual software modules are combined and tested as a group. Integration testing is conducted to evaluate the compliance of a system or component with specified functional requirements. It occurs after unit testing and before validation testing. Integration testing takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an integration test plan to those aggregates, and delivers as its output the integrated system ready for system testing[

TESTING FRAMEWORKS

JUNIT 5

DEFINITION

As per: https://junit.org/junit5/docs/current/user-guide/#overview-getting-started
Unlike previous versions of JUnit, JUnit 5 is composed of several different modules from three different sub-projects.
JUnit 5 = JUnit Platform + JUnit Jupiter + JUnit Vintage
The JUnit Platform serves as a foundation for launching testing frameworks on the JVM. It also defines the TestEngine API for developing a testing framework that runs on the platform. Furthermore, the platform provides a Console Launcher to launch the platform from the command line and a JUnit 4 based Runner for running any TestEngine on the platform in a JUnit 4 based environment. First-class support for the JUnit Platform also exists in popular IDEs (see IntelliJ IDEA, Eclipse, NetBeans, and Visual Studio Code) and build tools (see Gradle, Maven, and Ant).

POM DEPENDENCIES

```
<dependency>
	<groupId>org.junit.jupiter</groupId>
	<artifactId>junit-jupiter-api</artifactId>
	<version>${junit.version}</version>
	<scope>test</scope>
</dependency>

<dependency>
	<groupId>org.junit.vintage</groupId>
	<artifactId>junit-vintage-engine</artifactId>
	<version>${junit.version}</version>
	<scope>test</scope>
</dependency>

<dependency>
	<groupId>org.junit.jupiter</groupId>
	<artifactId>junit-jupiter-engine</artifactId>
	<version>${junit.version}</version>
</dependency>

<dependency>
	<groupId>org.junit.jupiter</groupId>
	<artifactId>junit-jupiter-params</artifactId>
	<version>${junit.version}</version>
	<scope>test</scope>
</dependency>
```

ASSERTJ

DEFINITION

As per: https://assertj.github.io/doc/#assertj-overview
AssertJ is a java library providing a rich set of assertions, truly helpful error messages, improves test code readability and is designed to be super easy to use within your favourite IDE.
http://www.javadoc.io/doc/org.assertj/assertj-core/ is the latest version of assertj core javadoc, each assertion is explained, most of them with code examples so be sure to check it if you want to know what a specific assertion does.
In summary this API gives more verbose assertion, can't be used simultaneously with JUnit

POM DEPENDENCIES

```
<dependency>
	<groupId>org.assertj</groupId>
	<artifactId>assertj-core</artifactId>
		<!-- use 2.9.1 for Java 7 projects -->
	<version>${assertj.version}</version>
	<scope>test</scope>
</dependency>
```

MOCKITO

DEFINITION

AS per : https://site.mockito.org/
Mockito is a mocking framework which lets you write beautiful tests with a clean & simple API. Mockito doesn't give you hangover because the tests are very readable and they produce clean verification errors.

POM DEPENDENCIES

```
<dependency>
	<groupId>org.mockito</groupId>
	<artifactId>mockito-junit-jupiter</artifactId>
	<version>3.8.0</version>
	<scope>test</scope>
</dependency>
```

UPDATE

JUNIT

To be compliant with the actual JUnit API we will update the project dependency JUnit4 to JUnit5.

ISSUE

When we have updated project dependency JUnit from JUnit4 to JUnit5, the issue was that the tests were not ran through maven test.
When use of *@ParametrizedTest* issue on *@ValueSymbol* caused by the absence of:
junit-jupiter-params dependency

SOLUTION

The solution was to add supplementary dependencies which were :

- junit-vintage-engine
- junit-jupiter-engine
- junit-jupiter-params

Moreover, the use of JUnit 5 leads to an update of maven-surefire-plugin at least version 2.22.2

AUDIT GENERAL APPROACH

PROJECT ANALYSIS

1- At reception run test
2- Correct issues if needed.
3- Update if necessary.
4- Adding useful and necessary dependencies
5- Check what is code against what is expected.
6- Code analysis.
7- Correction if needed.
8- Refactoring.
9- Improvement.
10- Functionality Implementation if needed.
11- Test Implementation if needed.
12- TODO task implementation.
13- Unit test on business(here means intelligence) implementation.
14- Integration test on business(here means intelligence) implementation.
15- Control coverage
16- Site generation.

As the analysis and testing is not a linear process, we might have to switch between the different steps in function of the eventual issues revealed during test.
Moreover, the Javadoc will be if needed corrected and implemented all along the process.

REFACTORING

The objective of the refactoring here is to made methods more “granular” i.e., one method one action (as far as possible), by this way it will be easier to perform testing, and in case of needs debugging.

So, a “big” method accomplishing several actions, will be decomposed on several small methods which accomplish one action, then all these new parts will be centralized in the big one.

This approach will improve code comprehension and traceability.

TESTING

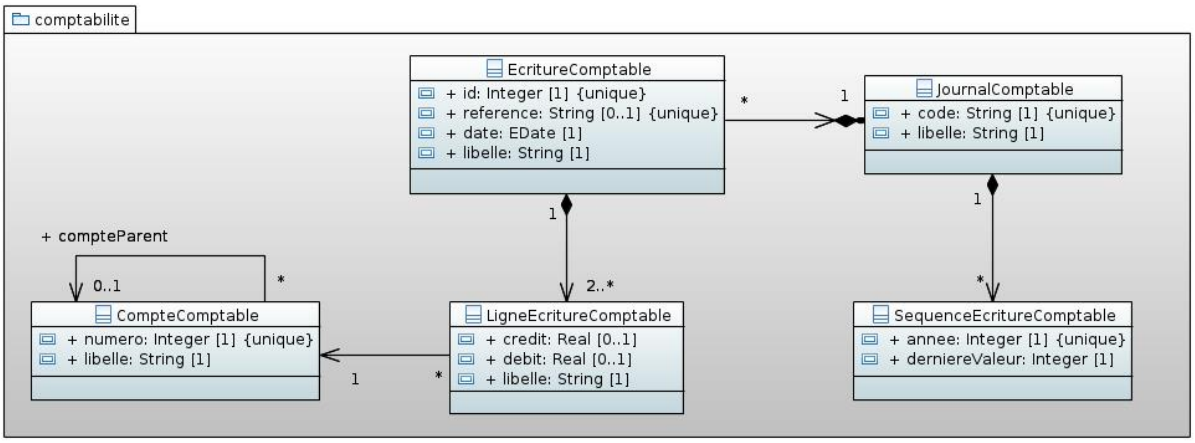
As far as possible the approach will be to create the object to test respecting all constrains and expectations, then vary one by one all argument, actors, conditions with a dedicated test for each of these elements.
After that we will test the methods behavior. And finally perform integration tests.
By this way we will maximize the test coverage and ease the debugging if needed.

PROJECT DEFINITION AND SPECIFICATION

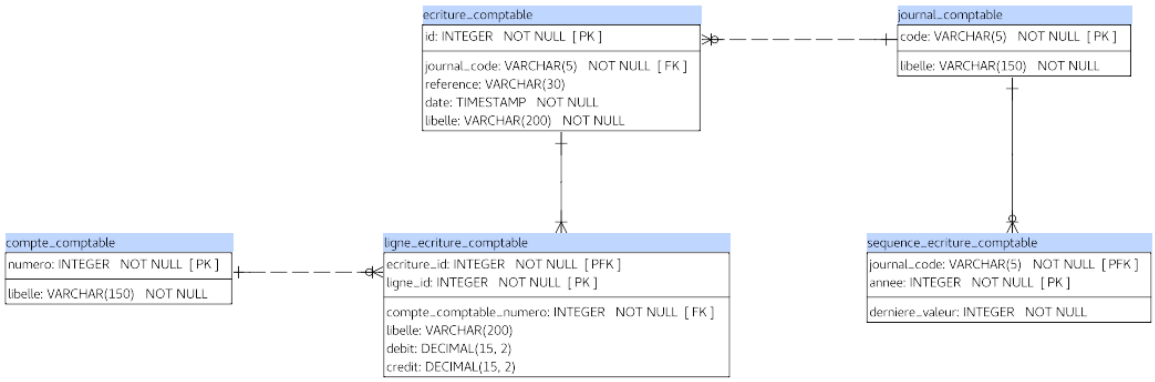
CLASS DIAGRAM AND PHYSICAL DATA MODEL

The following class diagram give the information concerning the class, the attributes, the constrains to be applied to attributes, and the relationship between actors.

CLASS DIAGRAM



PHYSICAL DATA MODEL



SPECIFICATIONS AND FUNCTIONNALITIES

The following list gives the API specification and functionalities determined by the client and the analyst programmer.

ID	Description
RG_Compta_1	Le solde d'un compte comptable est égal à la somme des montants au débit des lignes d'écriture diminuées de la somme des montants au crédit. Si le résultat est positif, le solde est dit "débiteur", si le résultat est négatif le solde est dit "créditeur".
RG_Compta_2	Pour qu'une écriture comptable soit valide, elle doit être équilibrée : la somme des montants au crédit des lignes d'écriture doit être égale à la somme des montants au débit.
RG_Compta_3	Une écriture comptable doit contenir au moins deux lignes d'écriture : une au débit et une au crédit.
RG_Compta_4	Les montants des lignes d'écriture sont signés et peuvent prendre des valeurs négatives (même si cela est peu fréquent).
RG_Compta_5	La référence d'une écriture comptable est composée du code du journal dans lequel figure l'écriture suivi de l'année et d'un numéro de séquence (propre à chaque journal) sur 5 chiffres incrémenté automatiquement à chaque écriture. Le formatage de la référence est : XX-AAAA/#####. Ex : Journal de banque (BQ), écriture au 31/12/2016 --> BQ-2016/00001
RG_Compta_6	La référence d'une écriture comptable doit être unique, il n'est pas possible de créer plusieurs écritures ayant la même référence.
RG_Compta_7	Les montants des lignes d'écritures peuvent comporter 2 chiffres maximum après la virgule.

FIRST BUILD

AIM

Before anything at reception of the project a first maven build is launched using command :

```
mvn clean install
```

To check if the project is clean with no major issues and if some testing methods are already implemented.

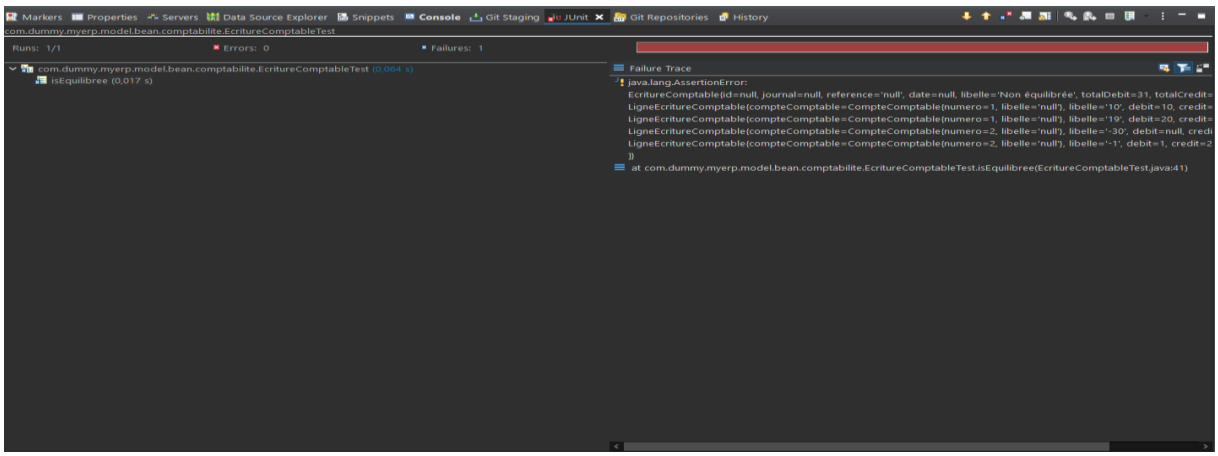
BUILD-1

RESULT

```
[INFO]
[INFO] Results:
[INFO]
[ERROR] Failures:
[ERROR]   EcritureComptableTest.isEquilibree:41 EcritureComptable(id=null, journal=null, reference='null', date=null, libelle='Non équilibrée', totalDebit=31, totalCredit=31, listLigneEcriture=[
LigneEcritureComptable(compteComptable=CompteComptable(numero=1, libelle='null'), libelle='10', debit=10, credit=null)
LigneEcritureComptable(compteComptable=CompteComptable(numero=1, libelle='null'), libelle='19', debit=20, credit=1)
LigneEcritureComptable(compteComptable=CompteComptable(numero=2, libelle='null'), libelle='-30', debit=null, credit=30)
LigneEcritureComptable(compteComptable=CompteComptable(numero=2, libelle='null'), libelle='-1', debit=1, credit=2)
])
[INFO]
[ERROR] Tests run: 1, Failures: 1, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] Reactor Summary for MyERP 1.x.x-SNAPSHOT:
[INFO]
[INFO] MyERP ..................................... SUCCESS [  0.002 s]
[INFO] myerp-technical ........................... SUCCESS [  0.824 s]
[INFO] myerp-model ............................... FAILURE [  1.114 s]
[INFO] myerp-consumer ............................ SKIPPED
[INFO] myerp-business ............................ SKIPPED
[INFO] ------------------------------------------------------------------------
[INFO] BUILD FAILURE
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  2.047 s
[INFO] Finished at: 2021-02-17T10:24:38+01:00
[INFO] ------------------------------------------------------------------------
[ERROR] Failed to execute goal org.apache.maven.plugins:maven-surefire-plugin:2.20:test (default-test) on project myerp-model: There are test failures.
[ERROR]
[ERROR] Please refer to O:\Project\P9\P9_FLE\P9_SourceCode\projet_B4_FR.zip_expanded\src\myerp-model\target\surefire-reports for the individual test results.
[ERROR] Please refer to dump files (if any exist) [date]-jvmRun[N].dump, [date].dumpstream and [date]-jvmRun[N].dumpstream.
[ERROR] -> [Help 1]
[ERROR]
[ERROR] To see the full stack trace of the errors, re-run Maven with the -e switch.
[ERROR] Re-run Maven using the -X switch to enable full debug logging.
[ERROR]
[ERROR] For more information about the errors and possible solutions, please read the following articles:
[ERROR] [Help 1] http://cwiki.apache.org/confluence/display/MAVEN/MojoFailureException
[ERROR]
[ERROR] After correcting the problems, you can resume the build with the command
[ERROR]   mvn <args> -rf :myerp-model
```

ERROR ANALYSIS

JUNIT TEST REPORT-1



ANALISYS-1

The method isEqilibree() from the class myerp-model.src.main.java.com.dummy.myerp.model.bean.comptabilite.LigneEcritureComptable return true instead of false.

```
    @Test
    public void isEquilibree() {
        EcritureComptable vEcriture;
        vEcriture = new EcritureComptable();

        vEcriture.setLibelle("Equilibrée");
        vEcriture.getListLigneEcriture().add(this.createLigne(1, "200.50",
        vEcriture.getListLigneEcriture().add(this.createLigne(1, "100.50",
        vEcriture.getListLigneEcriture().add(this.createLigne(2, null, "301
        vEcriture.getListLigneEcriture().add(this.createLigne(2, "40", "7")
        Assert.assertTrue(vEcriture.toString(), vEcriture.isEquilibree());

        vEcriture.getListLigneEcriture().clear();
        vEcriture.setLibelle("Non équilibrée");
        vEcriture.getListLigneEcriture().add(this.createLigne(1, "10", null
        vEcriture.getListLigneEcriture().add(this.createLigne(1, "20", "1")
        vEcriture.getListLigneEcriture().add(this.createLigne(2, null, "30"
        vEcriture.getListLigneEcriture().add(this.createLigne(2, "1", "2"))
        Assert.assertFalse(vEcriture.toString(), vEcriture.isEquilibree());
    }

}
```

INVESTIGATION-1

```
    /**
     * Renvoie si l'écriture est équilibrée (TotalDebit = TotalCrédit)
     * @return boolean
     */
    public boolean isEquilibree() {
        boolean vRetour = this.getTotalDebit().equals(getTotalCredit());
        return vRetour;
    }
```

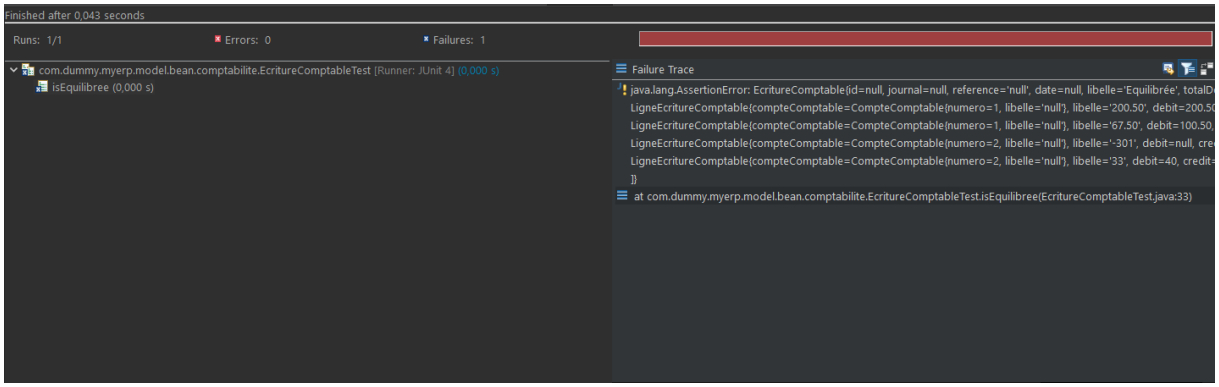
```
    public BigDecimal getTotalDebit() {
        BigDecimal vRetour = BigDecimal.ZERO;
        for (LigneEcritureComptable vLigneEcritureComptable : listLigneEcriture) {
            if (vLigneEcritureComptable.getDebit() != null) {
                vRetour = vRetour.add(vLigneEcritureComptable.getDebit());
            }
        }
        return vRetour;
    }
```

```java
public BigDecimal getTotalCredit() {
    BigDecimal vRetour = BigDecimal.ZERO;
    for (LigneEcritureComptable vLigneEcritureComptable : listLigneEcriture) {
        if (vLigneEcritureComptable.getDebit() != null) {
            vRetour = vRetour.add(vLigneEcritureComptable.getDebit());
        }
    }
    return vRetour;
}
```

CORRECTION-1

```java
public BigDecimal getTotalCredit() {
    BigDecimal vRetour = BigDecimal.ZERO;
    for (LigneEcritureComptable vLigneEcritureComptable : listLigneEcriture) {
        if (vLigneEcritureComptable.getCredit() != null) {
            vRetour = vRetour.add(vLigneEcritureComptable.getCredit());
        }
    }
    return vRetour;
}
```

JUNIT TEST REPORT-2



ANALISYS-2

The method isEqilibree() from the class myerp-model.src.main.java.com.dummy.myerp.model.bean.comptabilite.LigneEcritureComptable return false instead of true.

```java
@Test
public void isEquilibree() {
    EcritureComptable vEcriture;
    vEcriture = new EcritureComptable();

    vEcriture.setLibelle("Equilibrée");
    vEcriture.getListLigneEcriture().add(this.createLigne(1, "200.50", null));
    vEcriture.getListLigneEcriture().add(this.createLigne(1, "100.50", "33"));
    vEcriture.getListLigneEcriture().add(this.createLigne(2, null, "301"));
    vEcriture.getListLigneEcriture().add(this.createLigne(2, "40", "7"));
    Assert.assertTrue(vEcriture.toString(), vEcriture.isEquilibree());

    vEcriture.getListLigneEcriture().clear();
    vEcriture.setLibelle("Non équilibrée");
    vEcriture.getListLigneEcriture().add(this.createLigne(1, "10", null));
    vEcriture.getListLigneEcriture().add(this.createLigne(1, "20", "1"));
    vEcriture.getListLigneEcriture().add(this.createLigne(2, null, "30"));
    vEcriture.getListLigneEcriture().add(this.createLigne(2, "1", "2"));
    Assert.assertFalse(vEcriture.toString(), vEcriture.isEquilibree());
}
```

INVESTIGATION-2

```
/**
 * Renvoie si l'écriture est équilibrée (TotalDebit = TotalCrédit)
 * @return boolean
 */
public boolean isEquilibree() {
    boolean vRetour = this.getTotalDebit().equals(getTotalCredit());
    return vRetour;
}
```

```
public BigDecimal getTotalDebit() {
    BigDecimal vRetour = BigDecimal.ZERO;
    for (LigneEcritureComptable vLigneEcritureComptable : listLigneEcriture) {
        if (vLigneEcritureComptable.getDebit() != null) {
            vRetour = vRetour.add(vLigneEcritureComptable.getDebit());
        }
    }
    return vRetour;
}
```

```
public BigDecimal getTotalCredit() {
    BigDecimal vRetour = BigDecimal.ZERO;
    for (LigneEcritureComptable vLigneEcritureComptable : listLigneEcriture) {
        if (vLigneEcritureComptable.getCredit() != null) {
            vRetour = vRetour.add(vLigneEcritureComptable.getCredit());
        }
    }
    return vRetour;
}
```

No major issue, the problem here is on the scale format, indeed in the test the LigneEcritureCompta is set with a debit using a 2 digits fractionnal BigDecimal and credit using a 0 digits fractionnal BigDecimal.
The comparison between both are so false, because ef object.equals used with BigDecimal.
As per Javadoc:
.equals() = Compares this BigDecimal with the specified Object for equality. Unlike compareTo, this method considers two BigDecimal objects equal only if they are equal invalue and scale (thus 2.0 is not equal to 2.00 when compared bythis method).

CORRECTION-2

Add fixed scale number in LigneEcritureComptable

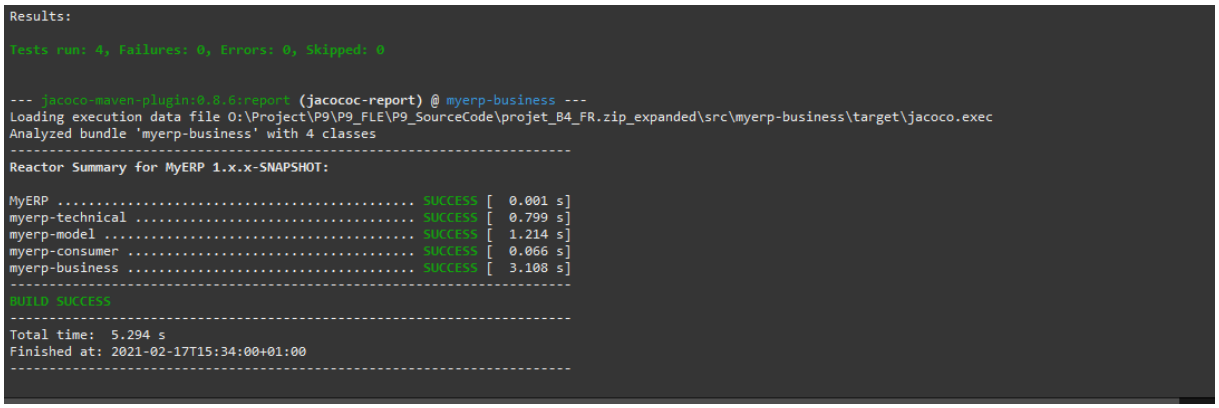
private static final int BIGDECIMAL_MIN_SCALE = 2;

and method

```
private BigDecimal fixedScale(BigDecimal pBigDecimal) {
                return pBigDecimal.setScale(BIGDECIMAL_MIN_SCALE);}
```

BUILD-2

RESULT



CONCLUSION

All builds get the status success, project clean and all clear for the next steps.

BEANS/MODELS CHECK

AIM

In a quality assurance process, it is advised to:
"Write what is done, do what is wrote".
The beans/models check will be performed on classes in charge of representing data base tables, here all those ones are centralized in the project module :

myerp-model.

This check will be processed in three steps :

- *analysis*
- *correction*
- *test*

ANALYSIS STEP

Check what is wrote: class Diagram and physical data model.
Against what is done: bean/model coded in java classes.

In two phases, first check if all attributes are present, then check individually each attributes.

The analysis will be performed according to the increasing complexity of the code, i.e. from primitive java type to custom classes, and from dependency to dependent i.e. if a class contains as attribute an another class, the attribute class will be analyzed first. Below the analysis order:

- CompteComptable.java
- JournalComptable.java
- SequenceEcritureComptable.java
- LigneEcritureComptable.java
- EcritureComptable.java

CORRECTION STEP

If errors are detected in the previous step, they will be corrected. For example, if String type is expected for an attribute and is set to Integer, the type will be modified in consequence.

TEST STEP

Concerning this step, we will not test java, i.e. all functionalities inherent to java for examples:

- Getter/Setter
- Annotation
- Type
- …

That why all constrains will be tested in customs functionalities / method calling java behavior, i.e.:

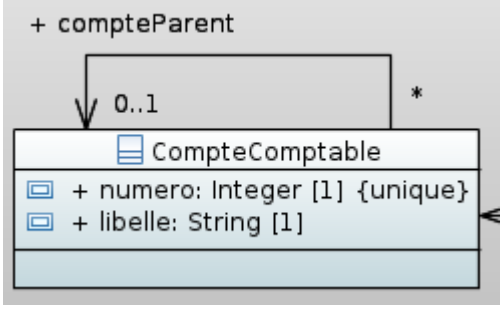
- a method coded to throw an exception if a constrain is not respected.
- we will test if this method well throws the expected exception.

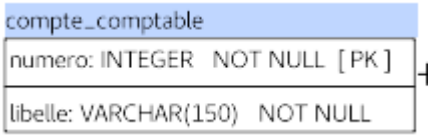
Note: the tests are performed in generally in the business layer, however if beans/models contain some method or functionalities they can be tested in the model layer, but it is not a particularly good practice.

MYERP-MODEL PROJECT MODULE

COMPTECOMPTABLE.JAVA

DEFINITION





SOURCE

com.dummy.myerp.model.bean.comptabilite.CompteComptable

```java
/**
 * Bean représentant un Compte Comptable
 */
public class CompteComptable {
    // ==================== Attributs ====================
    /** The Numero. */
    @NotNull
    private Integer numero;

    /** The Libelle. */
    @NotNull
    @Size(min = 1, max = 150)
    private String libelle;
```

ATTRIBUTES GENERAL CHECK

ANALYSIS

Expected	Present	result
numero	numero	
libelle	libelle	

CORRECTION

N/A

ATTRIBUTES INDIVIDUAL CHECK

ATTRIBUTE: NUMERO

Analysis

Type expected	Type assigned	result
Integer	Integer	

Constrains expected	Constrains assigned	results
Not null	*@NotNull*	
Unique	N/A*	

*Insert/create CompteComptable not implemented the unicity is supposed asserted by and in the data base, moreover no call to the data base at EcritureComptable insertion.

Correction

N/A

ATTRIBUTE: LIBELLE

Analysis

Type expected	Type assigned	result
String	String	

Constrains expected	Constrains assigned	results
Not null	*@NotNull*	
Size 1 to 150	*@Size(min=1, max=150)*	

Correction

N/A

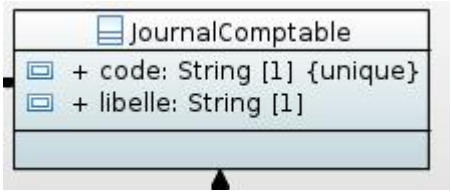
TEST

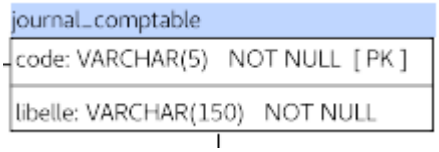
TEST CLASSES

com.dummy.myerp.model.bean.comptabilite.CompteComptableTest

JOURNALCOMPTABLE.JAVA

DEFINITION





SOURCE

com.dummy.myerp.model.bean.comptabilite.JournalComptable

```java
/**
 * Bean représentant un Journal Comptable
 */
public class JournalComptable {

    // ==================== Attributs ====================
    /** code */
    @NotNull
    @Size(min = 1, max = 5)
    private String code;

    /** libelle */
    @NotNull
    @Size(min = 1, max = 150)
    private String libelle;
```

ATTRIBUTES GENERAL CHECK

ANALYSIS

Expected	Present	result
code	numero	
libelle	libelle	

Correction

N/A

ATTRIBUTES INDIVIDUAL CHECK

ATTRIBUTE: CODE

Analysis

Type expected	Type assigned	result
String	String	

Constrains expected	Constrains assigned	results
Not null	*@NotNull*	
Size 1 to 5	*@Size(min=1, max=5)*	

Correction

N/A

ATTRIBUTE: LIBELLE

Analysis

Type expected	Type assigned	result
String	String	

Constrains expected	Constrains assigned	results
Not null	*@NotNull*	
Size 1 to 150	*@Size(min=1, max=150)*	

Correction

N/A

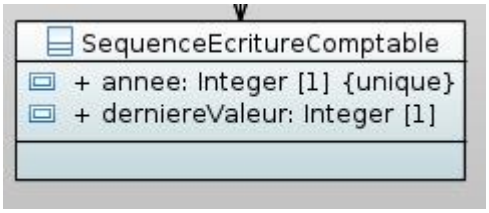
TEST

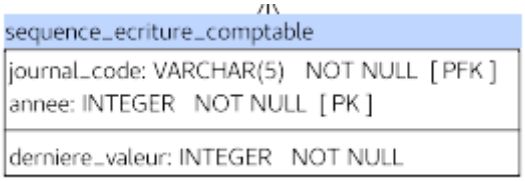
TEST CLASSES

com.dummy.myerp.model.bean.comptabilite.JournalComptableTest
com.dummy.myerp.business.impl.manager.ComptabiliteManagerImplTest

SEQUENCEECRITURECOMPTABLE.JAVA

DEFINITION





SOURCE

com.dummy.myerp.model.bean.comptabilite.SequenceEcritureComptable

```java
/**
 * Bean représentant une séquence pour les références d'écriture comptable
 */
public class SequenceEcritureComptable {

    // ==================== Attributs ====================
    /** L'année */
    private Integer annee;
    /** La dernière valeur utilisée */
    private Integer derniereValeur;
```

ATTRIBUTES GENERAL CHECK

ANALYSIS

Expected	Present	result
annee	annee	
dernieReValeur	dernieReValeur	
journalComptabe (induced by PDM)	N/A	

CORRECTION

Add JournalCompatble journalComptable attribute.

Expected	Present	result
journalComptabe (induced by PDM)	journalComptable	

ATTRIBUTES INDIVIDUAL CHECK

ATTRIBUTE: ANNEE

Analysis

Type expected	Type assigned	result
Integer	Integer	

Constrains expected	Constrains assigned	results
Not null	N/A	
Unique	N/A*	

Why it is set to unique in Class Diagram? As there is multiple JournalComptable in a year, and the relation ship is set to a one-to-many between JournalComptable and SequenceEcritureComptable

Correction

Add *@NotNull* annotation

Constrains expected	Constrains assigned	results
Not null	*@NotNull*	

ATTRIBUTE: DERNIEREVALEUR

Analysis

Type expected	Type assigned	result
Integer	Integer	

Constrains expected	Constrains assigned	results
Not null	N/A	

Correction

Add *@NotNull* annotation

Constrains expected	Constrains assigned	results
Not null	*@NotNull*	

ATTRIBUTE: JOURNALCOMPTABLE

Analysis

Type expected	Type assigned	result
JournalComptable	JournalComptable r	

Constrains expected	Constrains assigned	results
Not null	*@NotNull* l	

Correction

N/A

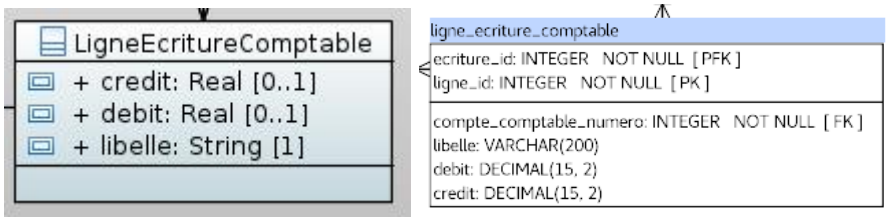
TEST

TEST CLASSES

com.dummy.myerp.model.bean.comptabilite.SequenceEcritureComptableTest
com.dummy.myerp.business.impl.manager.ComptabiliteManagerImplTest

LIGNEECRITURECOMPTABLE.JAVA

DEFINITION



SOURCE

com.dummy.myerp.model.bean.comptabilite.LigneEcritureComptable

```java
/**
 * Bean représentant une Ligne d'écriture comptable.
 */
public class LigneEcritureComptable {

    /** Compte Comptable */

    @NotNull
    private CompteComptable compteComptable;

    /** The Libelle. */
    @Size(max = 200)
    private String libelle;

    /** The Debit. */
    @MontantComptable
    private BigDecimal debit;

    /** The Credit. */
    @MontantComptable
    private BigDecimal credit;
```

ATTRIBUTES GENERAL CHECK

ANALYSIS

Expected	Present	result
credit	credit	
debit	debit	
libelle	libelle	
CompteComptabe (induced by PDM)	compteComptable	

CORRECTION

N/A

ATTRIBUTES INDIVIDUAL CHECK

ATTRIBUTE: CREDIT

Analysis

Type expected	Type assigned	result
Real	BigDecimal	

Constrains expected	Constrains assigned	results
RG_Compta_7	*@MontantComptable*	

Correction

N/A

ATTRIBUTE: DEBIT

Analysis

Type expected	Type assigned	result
Real	BigDecimal	

Constrains expected	Constrains assigned	results
RG_Compta_7	*@MontantComptable*	

Correction

N/A

ATTRIBUTE: LIBELLE

Analysis

Type expected	Type assigned	result
String	String	

Constrains expected	Constrains assigned	results
Not null	N/A	
Size 1 to 200	*@Size(max = 200)*	

Correction

Add *@NotNull* annotation
Correct *@Size*.

Constrains expected	Constrains assigned	results
Not null	*@NotNull*	
Size 1 to 200	*@Size(min = 1, max = 200)*	

ATTRIBUTE: COMPTECOMPTABLE

Analysis

Type expected	Type assigned	result
CompteComptable	CompteComptable	

Constrains expected	Constrains assigned	results
Not null	*@NotNull*	

Correction

N/A

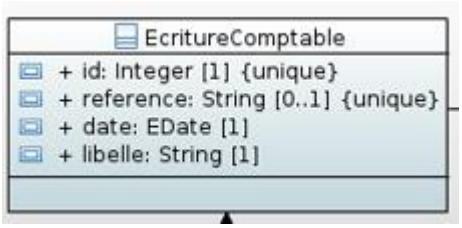
TEST

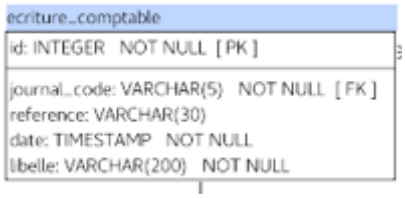
TEST CLASSES

com.dummy.myerp.model.bean.comptabilite.LigneEcritureComptableTest
com.dummy.myerp.business.impl.manager.ComptabiliteManagerImplTest

ECRITURECOMPTABLE.JAVA

DEFINITION





SOURCE

com.dummy.myerp.model.bean.comptabilite.EcritureComptable.class

```
public class EcritureComptable {

    // ==================== Attributs ====================
    /** The Id. */
    private Integer id;
    /** Journal comptable */
    @NotNull
    private JournalComptable journal;
    /** The Reference. */
    @Pattern(regexp = "\\d{1,5}-\\d{4}/\\d{5}")
    private String reference;
    /** The Date. */
    @NotNull
    private Date date;

    /** The Libelle. */
    @NotNull
    @Size(min = 1, max = 200)
    private String libelle;

    /** La liste des lignes d'écriture comptable. */
    @Valid
    @Size(min = 2)
    private final List<LigneEcritureComptable> listLigneEcriture = new ArrayList<>();
```

ATTRIBUTES GENERAL CHECK

ANALYSIS

Expected	Present	result
id	credit	
reference	debit	
date	libelle	
libelle	CompteComptable	
List LigneEcritureCompatble (induced by PDM)	listLigneEcritureComptable	
JournalComptable (induced by PDM)	journalComptable	

CORRECTION

N/A

ATTRIBUTES INDIVIDUAL CHECK

ATTRIBUTE: ID

Analysis

Type expected	Type assigned	result
Integer	Integer	

Constrains expected	Constrains assigned	results
Not null	N/A	
Unique	N/A*	

*Unicity constrain managed by:
com.dummy.myerp.consumer.dao.impl.db.dao.ComptabiliteDaoImpl

Correction

Attribute id -> add *@NotNull*

Constrains expected	Constrains assigned	results
Not null	*@NotNull*	
Unique	N/A*	

ATTRIBUTE: REFERENCE

Analysis

Type expected	Type assigned	result
Integer	Integer	

Constrains expected	Constrains assigned	results
Regex Pattern AA-YYYY/xxxxxx	“\\d{1,5}-\\d{4}/\\d{5}”	

Correction

Regarding the relationship with JournalComptable and the RG_Compta_5 issue detected on regex which is*: “\\d{1,5}-\\d{4}/\\d{5}”.*

\\d{1,5} at the regex beginning corresponding to the journal code is no compliant with the definition in the JournalComptable classe, that wait for a String type.
The rest of the regex is correct and accepted.

Constrains expected	Constrains assigned	results
Regex Pattern AA-YYYY/xxxxxx	“[A-Z]{1,5}-\\d{4}/\\d{5}”	

Attribute: date

Analysis

Type expected	Type assigned	result
Date	Date	

Constrains expected	Constrains assigned	results
Not null	*@NotNull*	

Correction

N/A

Attribute: libelle

Analysis

Type expected	Type assigned	result
String	String	

Constrains expected	Constrains assigned	results
Not null	*@NotNull*	
Size 1 to 200	*@Size(min=1, max=200)*	

Correction

N/A

Attribute: journalComptable

Analysis

Type expected	Type assigned	result
JournalComptable	JournalComptable	

Constrains expected	Constrains assigned	results
Not null	*@NotNull*	

Correction

N/A

Attribute: listeLigneEcritureComptable

Analysis

Type expected	Type assigned	result
List<LigneEcritureComptable>	List<LigneEcritureComptable>	

Constrains expected	Constrains assigned	results
Not null	*@NotNull*	
Size 2 min	*@Size(min=2)*	
Valid* (induced by PDM)	*@Valid*	

*As insert EcritureComptable will insert LigneEcritureComptable too, that induced a valid element.

Correction

N/A

TESTING PHASE AND TODO TASKS

MYERP PARENT

TODO TASKS

N/A

UNIT TESTS CLASSES

See myerp-business and myerp-model

INTEGRATION TESTS CLASSES

See myerp-business

MYERP-TECHNICAL MODULE

TODO TASKS

N/A

TESTS CLASSES

See myerp-business

INTEGRATION TESTS CLASSES

See myerp-business

MYERP-CONSUMER MODULE

TODO TASKS

N/A

TESTS CLASSES

See myerp-business

MYERP-MODEL MODULE

TODO TASKS

// TODO à tester
public BigDecimal getTotalDebit()
Done
// TODO à tester
public BigDecimal getTotalCredit()
Done

UNIT TESTS CLASSES

src.test.java.com.dummy.myerp.model.bean.comptabilite.CompteComptableTest

src.test.java.com.dummy.myerp.model.bean.comptabilite.EcritureComptableTest

src.test.java.com.dummy.myerp.model.bean.comptabilite.JournalComptableTest

src.test.java.com.dummy.myerp.model.bean.comptabilite.LigneEcritureComptableTest

src.test.java.com.dummy.myerp.model.bean.comptabiliteSequenceEcritureComptableTest

**myerp-business.src.test.java.com.dummy.myerp.business.
dao.impl.db.dao.ComptabiliteDaoImpl. ComptabiliteManagerImplTest**

**myerp-business.src.test.java.com.dummy.myerp.business.
dao.impl.db.dao.ComptabiliteDaoImpl. ComptabiliteManagerImplMockTest**

INTEGRATION TESTS CLASSES

See myerp-business

MYERP-BUSINESS MODULE

TODO TASKS

// TODO ===== RG_Compta_5 : Format et contenu de la référence
// vérifier que l'année dans la référence correspond bien à la date de l'écriture, idem pour le code journal...
Done
// TODO à tester
    @Override
    public synchronized void addReference(EcritureComptable pEcritureComptable)
Done
// TODO à tester
    @Override
    public void checkEcritureComptable(EcritureComptable pEcritureComptable)
Done
// TODO tests à compléter
    protected void checkEcritureComptableUnit(EcritureComptable pEcritureComptable)
Done

UNIT TESTS CLASSES

**com.dummy.myerp.business.
dao.impl.db.dao.ComptabiliteDaoImpl. ComptabiliteManagerImplTest**

**com.dummy.myerp.business.
dao.impl.db.dao.ComptabiliteDaoImpl. ComptabiliteManagerImplMockTest**

INTEGRATION TESTS CLASSES

Note: in order to perform integration test a source folder IT has been added and all necessary source implemented :

- bootstrapContext
- transactionContext
- sqlContext
- PostgreSql database (as EcritureComptable id implementation is postgreSQl specific)

myerp-business.src.IT.java.com.dummy.myerp.business.dao.impl.db.dao.ComptabiliteDaoImpl.InitSpringInTest

APACHE MAVEN TEST

UNIT TEST

On each project

```
mvn clean test
```

Used on GitLAb :

```
mvn verify
```

INTEGRATION TEST

On MyERP

```
mvn test -Ptest-business
```

COVERAGE

IMPORTANT

In order to get all the project coverage a new model has been implemented :

- myerp-coverage

It is set with all project as depencencies and a profil to have only unit test coverage for myerp-business and myerp-model for GitLab, and all project for coverage on local. This is due to the specific postgreSQlL of the project and the fact that no postgreSql server are available on this project.
This new project is on charge to control the 75% global coverage.

RESULTS

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
myerp-consumer		82 %		50 %	14	74	39	236	8	64	1	13
myerp-model		79 %		40 %	32	98	41	204	4	61	0	5
myerp-technical		43 %		n/a	8	14	14	26	8	14	1	4
myerp-business		97 %		89 %	9	92	7	217	3	60	0	4
Total	378 of 2 609	85 %	61 of 158	61 %	63	278	101	683	23	199	2	26

Coverage Pass

SITE GENERATION

Use the maven commande :

```
mvn clean verify site site:deploy -P test-business,full-test-coverage
```

To generate the site, this one will have on more than the usual:

- Final Coverage report containing the global jacoco report
- A jacoco report for each modules
- This report as documentations

The site is at disposal in
${project.dir}/src/target/index.html

THE 4 ERRORS

Only for P9 Openclassrooms presentation and defense

➔Pattern regex on reference
➔Credit/debit on equilibree
➔scale bigdecimal on equilibree
➔Check rg compta 3 Nbrcredit < 1 instead of >=1 idem debit
Issue on integration test folder named as tes-business however convention impose that folder named

CONCLUSION

RG_COMPTA

RG_Compta n°	Covered	Result
1	Yes, by Bigdecimal	
2	Yes, method isEquilibree()	
3	Yes, methods in ComptabiliteManager	
4	Yes, by Bigdecimal	
5	Yes, methods in ComptabiliteManager	
6	Yes, methods in ComptabiliteManager plus contextSQL update with all needed method	
7	Yes, custom annotation *@MontantComptable*	

RG_Compta_1 no tested as no method implemented yet for getSolde(), however and same for RG_Compta_4, the Bigdecimal type allow and use signed numbers.

As we can see all RG_Compta are implemented.

TESTING AND COVERAGE

With a total of 144 tests and a global coverage at 85 % including :

- 79 % for myerp-model
- 97 % for myerp-business
- 82 % for myerp-consumer

With only tests put in place in myerp-business and myerp-model and using maven profile test-business and full-test-coverage, there is no need to add supplementary test for myerp-consummer and use profile test-consumer.
Concerning myerp-technical as it is mostly for exception purpose, the 43 % are accepted.

GLOSSARY

CD	Class Diagram
PDM	Physical Data Model
POM	Project Object Model
API	Application Programming Interface
IDE	Integrated development environment
N/A	Not Applicable
	Pass
	Fail
	Cannot status on it
	Pass after correction
@NotNull	javax.validation.constraints.NotNull
@Valid	javax.validation.Valid
@Size	javax.validation.constraints.Size
@MontantComptable	Custom annotation myerp-model. com.dummy.myerp.model.validation.constraint. MontantComptable.