

# Introducció a Concordion amb IntelliJ

## 1. Introducció

Concordion és un framework orientat al desenvolupament de tests d'acceptació. Funciona a través de tests escrits en HTML o mitjançant Markdown Language (a partir de la versió 2.0) que al seu torn són instrumentalitzats amb atributs especials que el framework interpreta per executar aquests tests.

En comptes de forçar que els Product Owners hagin d'especificar els requisits en un llenguatge amb una determinada estructura, Concordion permet definir-los utilitzant llenguatge natural, podent utilitzar paràgrafs, taules, i signes de puntuació. Això fa que les especificacions siguin molt més llegibles i senzilles d'escriure, a més d'ajudar enormement a la compressió i acceptació del que se suposa ha de fer la funcionalitat que anem a provar.

Les llibreries .jar de Concordion es poden descarregar aquí:

<http://concordion.org/download/java/markdown/>

## 2. Fent servir servir Concordion amb IntelliJ

### 2.1 – Conceptes clau

Cada característica o comportament s'ha d'especificar, implementar i verificar mitjançant les especificacions actives i la seva connexió amb el sistema en desenvolupament. Una especificació activa a Concordion consisteix en dues parts clau:

1. Un document de requisits ben escrit que descriu la funcionalitat desitjada (HTML). Les especificacions HTML contenen descripcions de la funcionalitat il·lustrada amb exemples de prova d'acceptació. Les dades d'exemples es marquen mitjançant etiquetes HTML.
2. Les proves d'acceptació s'escriuen en Java i s'anomena codi de fixació o «fixture code». Les proves es codifiquen implementant una extensió de Concordion d'un cas estàndard de Junit test case. El codi de fixació troba les dades d'exemple marcats per etiquetes i els utilitza per verificar el sistema en desenvolupament.

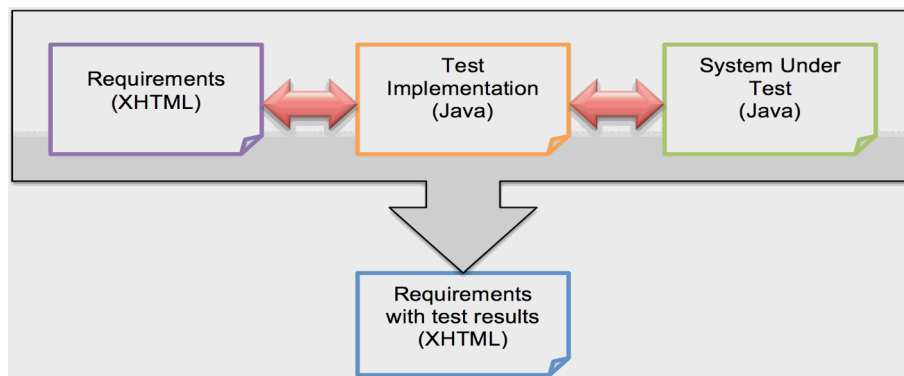


Figura 1: Concepte d'especificació activa.

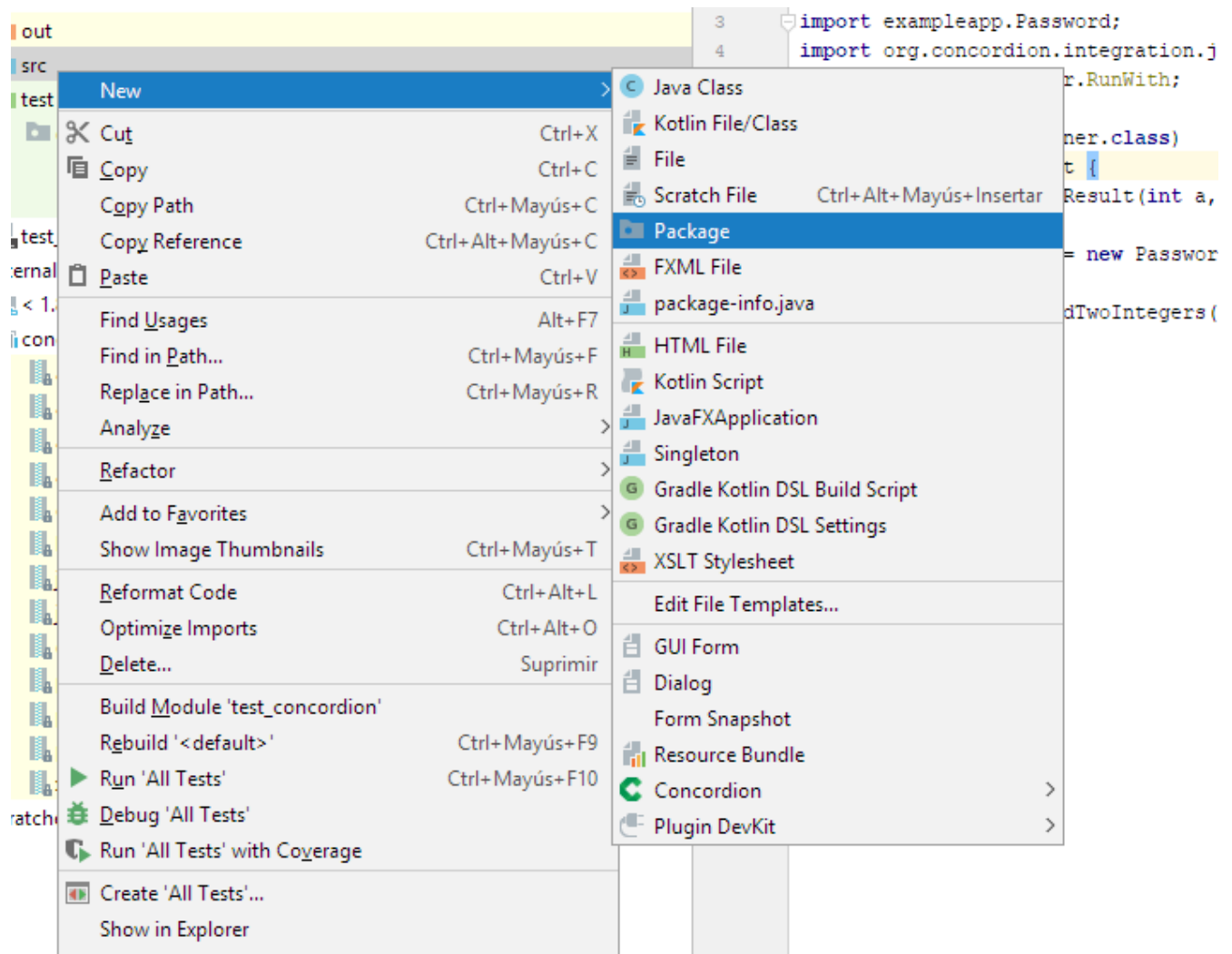
El concepte d'ús de Concordion s'il·lustra en el diagrama de la Figura 1. Les proves d'acceptació s'especifiquen utilitzant el llenguatge nadiu i s'organitzen en els fitxers Requirements (HTML). Les proves s'implementen en Java i connecten els exemples dels requisits amb el codi Java del sistema que s'està provant. El codi de prova està connectat amb els requisits mitjançant etiquetes HTML, que contenen ordres per Concordion. La realització de proves en Concordion dona com a resultat resultats de fitxers HTML que combinen els resultats de l'especificació i la prova originals. Les proves correctes es ressalten com a "verd" i no les que no tenen èxit amb "vermell". Aquest concepte de Concordion s'anomena especificacions actives (Active Specifications) a causa del fet que l'implementació de la prova vincula les especificacions amb el sistema. Qualsevol canvi en el sistema donarà lloc a proves fallides, el que ens

recordarà que hem d'anar actualitzant les especificacions. D'aquesta manera les especificacions mai es faran obsoletes.

Per tal que Concondion funcioni, l'estructura de fitxers del vostre projecte ha de seguir les certes regles. En l'exemple que es mostra a la Figura 2 organitzem especificacions i provem els fitxers d'implementació en una estructura de paquets, aquí anomenada **exampleapp.spec**.

### 2.1.1 – Dummy example: Crear un projecte a IntelliJ amb Concondion.

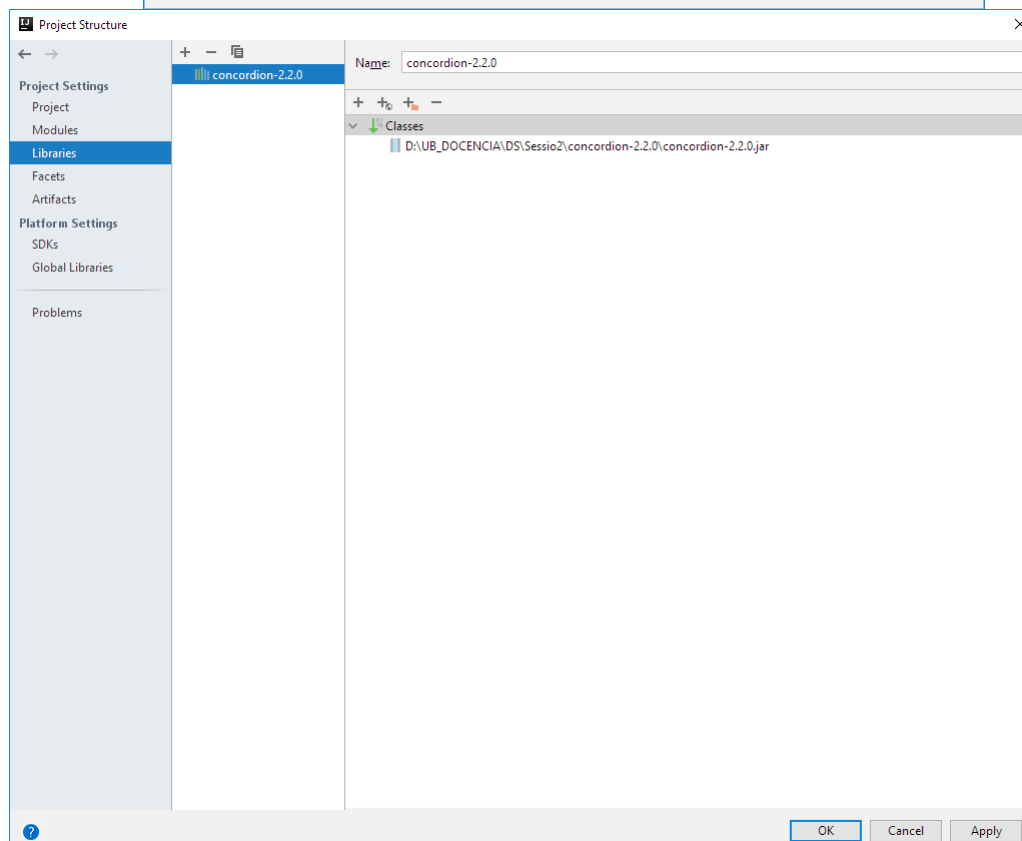
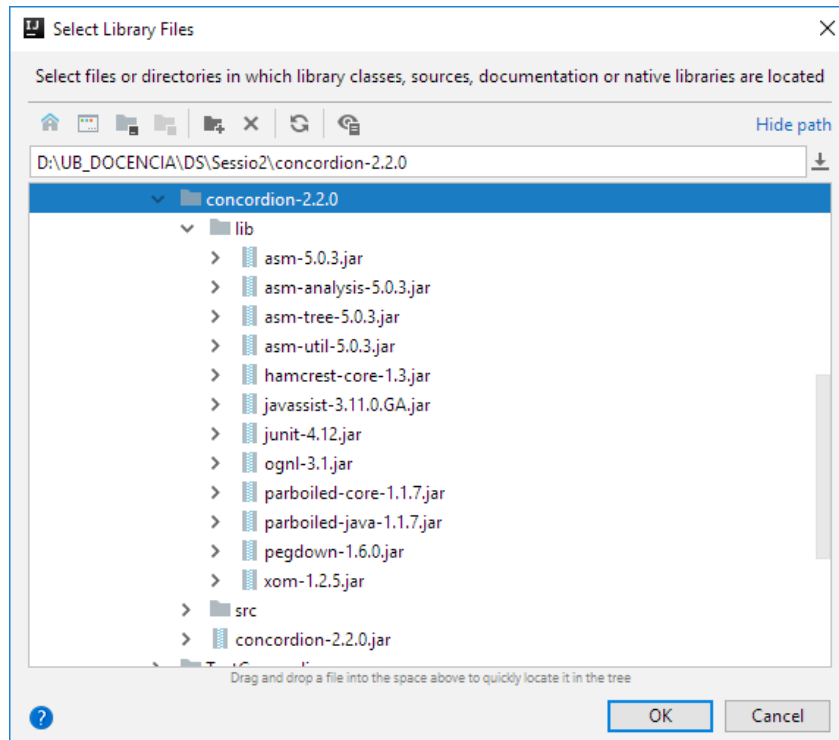
- Crear un nou projecte amb IntelliJ de la mateixa forma que hem vist en el document de la sessió 1 per a crear projectes d'aquest IDE.
- Dins del directori src generem un nou Package (per exemple: exampleapp):



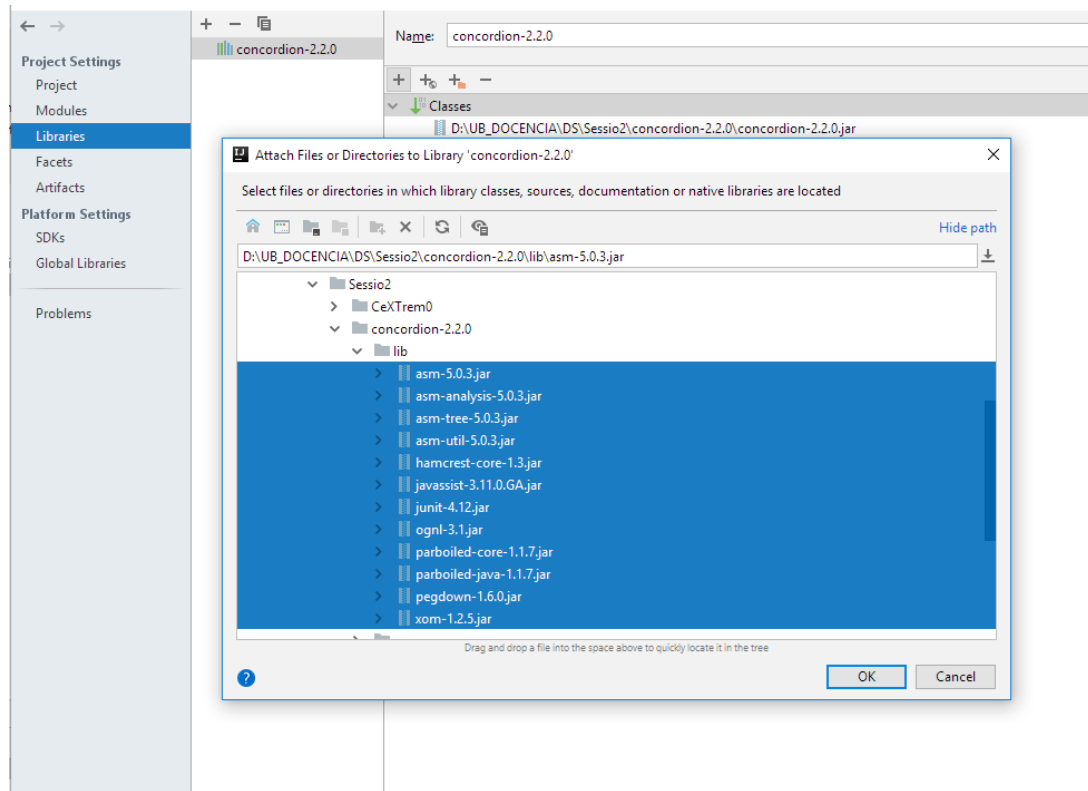
- Anem a fer el test d'acceptació:  
En cas que [context] quan [event] el sistema [resultat]

**Test d'acceptació:** US1.1 Suma de dos enters  
**En cas que** l'usuari entri dos valors x i y  
**Quan** premi <intro>  
**el sistema** donarà com a resultat per pantalla la suma de x i y

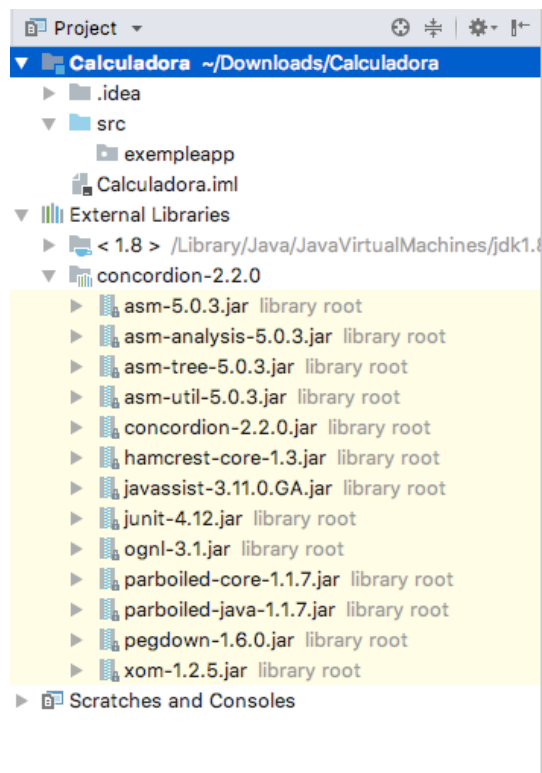
- Per això és necessari l'ús de Concordion
- Ara importarem les llibreries necessàries per al projecte. El plug-in de Concordion per a IntelliJ només us dona suport al implementar i crear les seves classes. Definirem a (*File* → *Project Structure* → *Libraries*) la ubicació física de la llibreria de Concordion, així com les dependències que tingui.
- Afegim primer el Jar de Concordion:



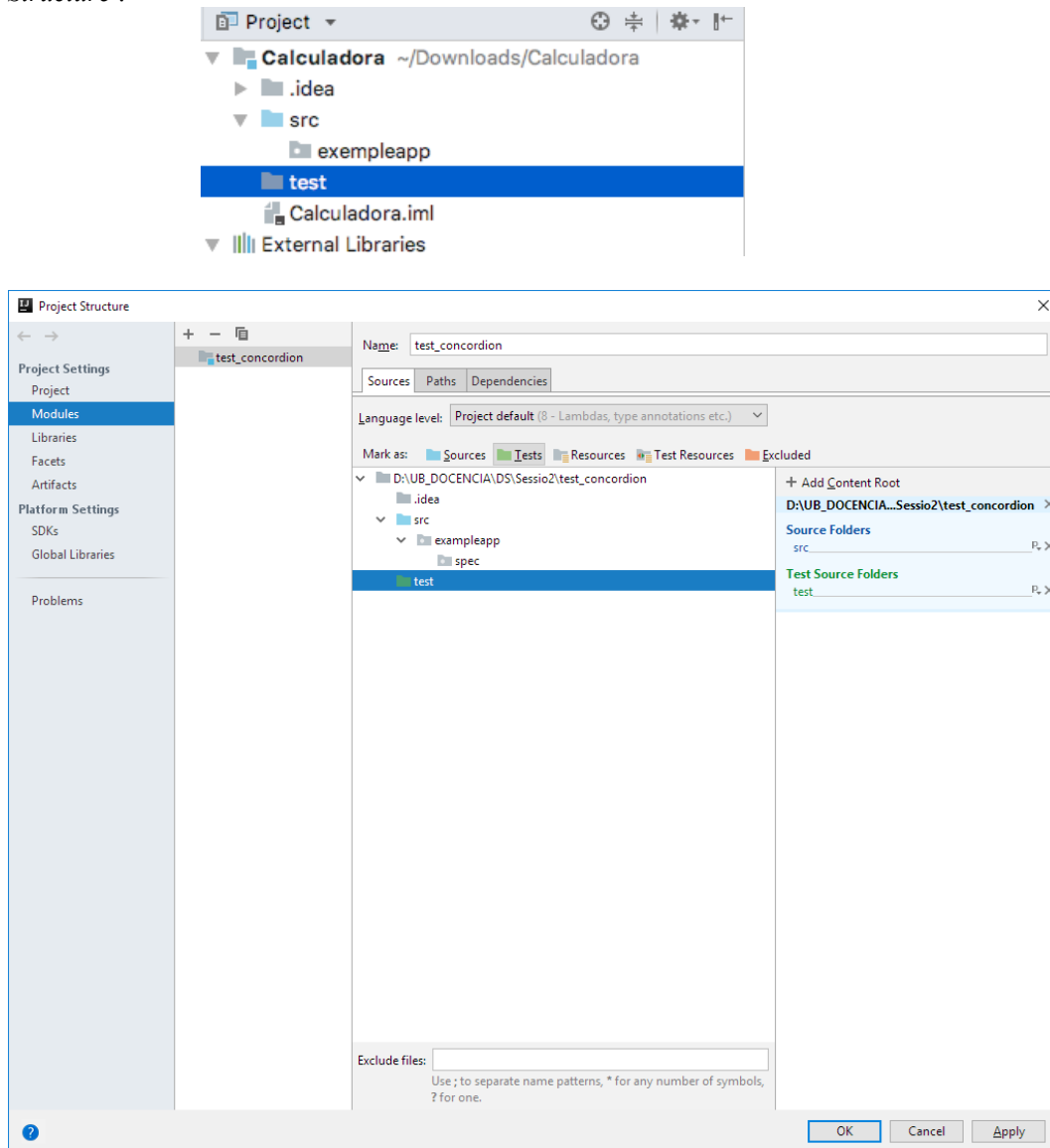
- Ara que ja tenim la llibreria de Concondion afegirem totes les llibreries dependents des de la finestra de la dreta (on apareix Classes).



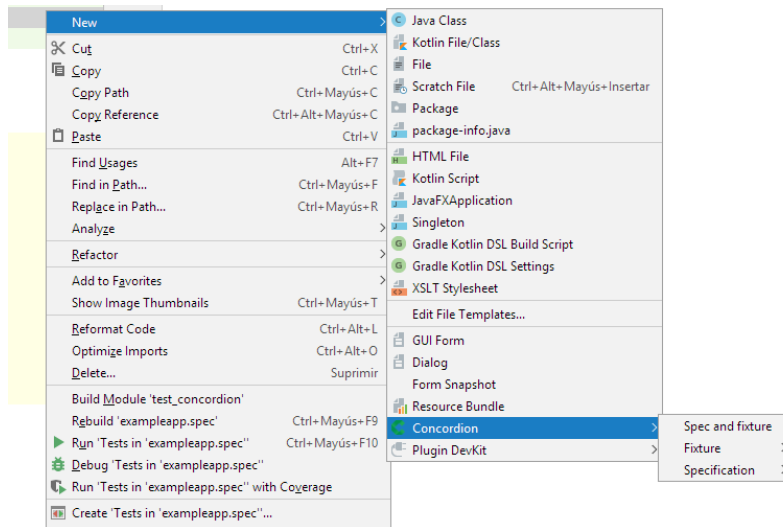
- Si repassem l'External libraries al projecte hauria d'aparèixer així:



- A continuació, per crear el primer test, creem un directori de test i el configurem com a tal al *Project Structure*.



- Crearem un Package dins d'aquest directori (Per exemple: exempleapp.spec) i ens ajudarem del plug-in de Concordion per IntelliJ per a crear els *Fixture* i *Spec* a dins. Cal baixar-lo de <https://plugins.jetbrains.com/plugin/7978-concordion-support> i després anar a File -> Settings -> Plugins i llegir-lo de disc (Install plugin from disk...) o bé baixar-lo directament del MarketPlace "Concordion support".



- Crearem primer un Specification anomenat Config.html que contindrà el següent codi:

```
<html xmlns:concordion="http://www.concordion.org/2007/concordion">
<head>
  <title>Config</title>
</head>
<body>
<h1>REQ-1.1 Add two integers</h1>
<p>
  Provides feature for adding two integers. For example
  <span concordion:set="#a">2</span> plus
  <span concordion:set="#b">5</span> should give
  <span concordion:assertEquals="getTestResult(#a,#b)">7</span>.
</p>
</body>
</html>
```

- Finalment crearem el fixture ConfigTest per assegurar-nos que hem configurat Concordion correctament al projecte. El contingut del codi serà el següent:

```
package exampleapp.spec;

import org.concordion.integration.junit4.ConcordionRunner;
import org.junit.runner.RunWith;

@RunWith(ConcordionRunner.class)
public class ConfigTest {
    public int getTestResult(int a, int b) {
        Calculadora calc = new Calculadora();
        return (calc.addTwoIntegers(a, b));
    }
}
```

- Ara cal posar la funcionalitat en la nostra aplicació de sumar dos enters. En el package dels fonts hem d'afegir el codi per a sumar dos enters:

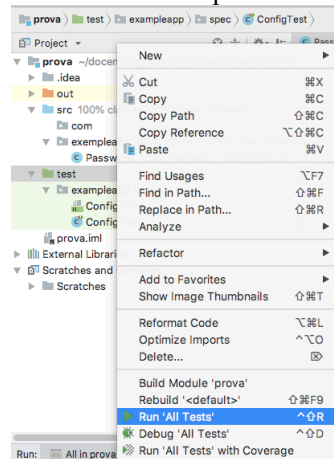
```
package exempleapp;

public class Calculadora {
    public int addTwoIntegers(int x, int y) {
        return x+y;
    }
}
```

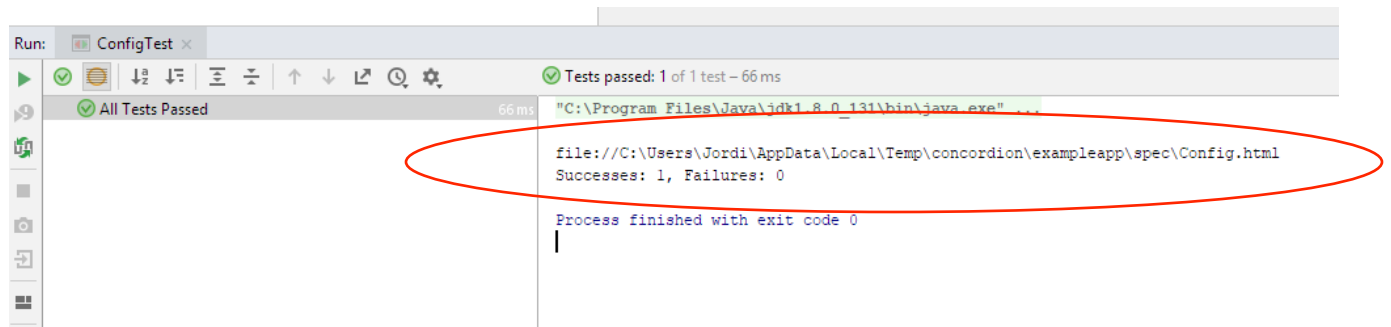
- I fer l'import a la classe Configuration.java de la classe que s'usarà, en aquest cas, la classe Calculadora:

```
import exempleapp.Calculadora;
```

- Tingueu en compte que us faran falta un sèrie d'imports per a que la classe compili, així també agafeu pràctica amb l'IDE i les funcions d'autoemplenar. Per executar els tests podeu seleccionar:



- El resultat al córrer els test serà:



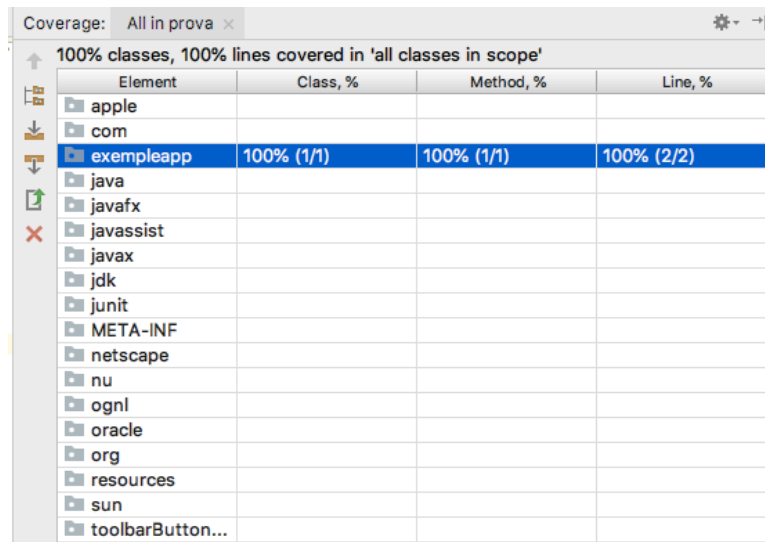
Si obrim l'enllaç html que ens donen els test, podem veure'ls en format html en un navegador.

En aquest cas (<file:///C:/Users/Jordi/AppData/Local/Temp/concordion/exampleapp/spec/Config.html>)

## REQ-1.1 Add two integers

Provides feature for adding two integers. For example 2 plus 5 should give 7.

També es pot seleccionar "Run All Test with Coverage" i dóna una taula amb el tant per cent del codi analitzat pels tests:



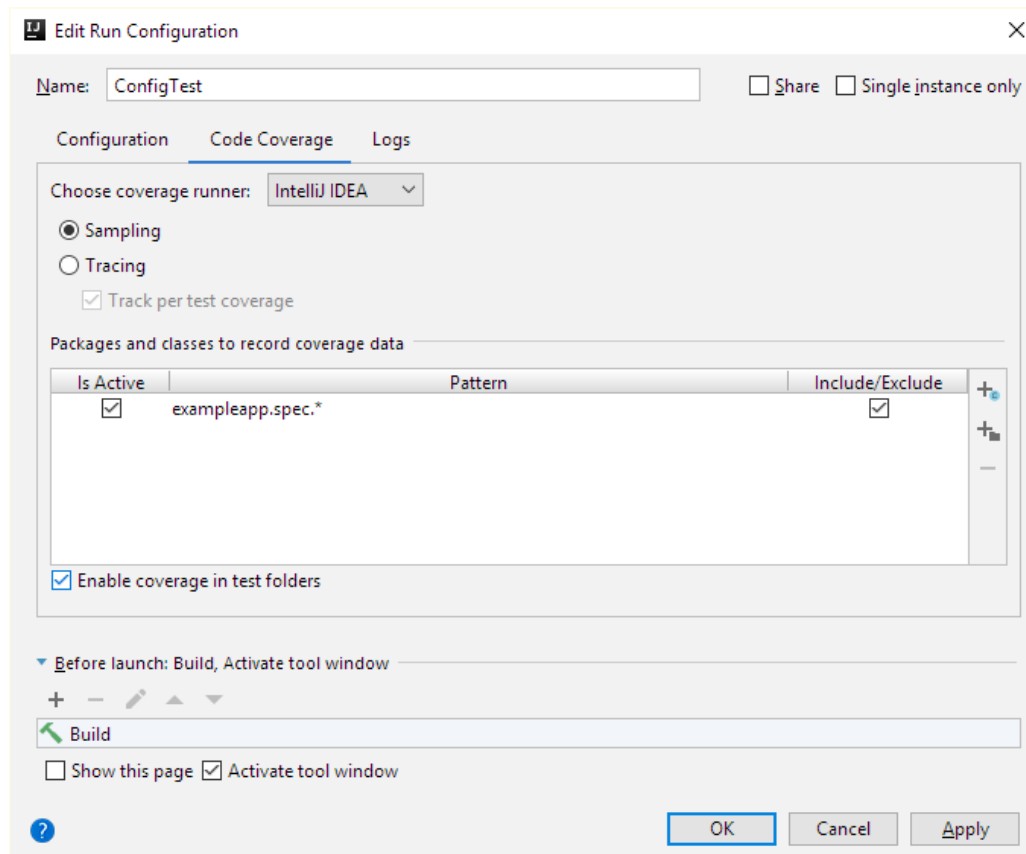
100% classes, 100% lines covered in 'all classes in scope'				
Element	Class, %	Method, %	Line, %	
apple				
com				
exempleapp	100% (1/1)	100% (1/1)	100% (2/2)	
java				
javafx				
javassist				
javax				
jdk				
junit				
META-INF				
netscape				
nu				
ognl				
oracle				
org				
resources				
sun				
toolbarButton...				

Si al córrer el test *with Coverage* reveu el missatge següent caldrà una petita configuració addicional al projecte.

No coverage results. Click [Edit](#) to fix configuration settings.

Si feu clic al enllaç *Edit* se us obrirà un menú, i anant a la pestanya *Code Coverage* haureu d'habilitar la opció *Enable coverage in test folders*.





## 2.2 – Exemple: Projecte CexTrem0

Descarreguem el projecte CeXTrem0 del Campus i l'importem a IntelliJ. Caldrà configurar les llibreries com en l'apartat anterior. Provem a executar el test i comprendre els resultats obtinguts. Modifiquem-lo per a fer els test d'acceptació de la història d'usuari explicada a les transparències de la sessió, on cal fer el registre d'un usuari.

## 3. Referències i més documentació.

<http://concordion.org/tutorial/java/html/>

<http://www.methodsandtools.com/tools/concordion.php>

<https://plus.google.com/+ConcordionOrganization>

<https://www.adictosaltrabajo.com/tutoriales/primeros-pasos-con-concordion/>

## 4. Exercici:

Realitzar l'història d'usuari d'enregistrar-se implementant-la després amb Concordion sobre l'exemple CeXTrem (<https://classroom.github.com/g/u-XOs9DU>)