**DOCUMENTO JUNIT**

Preambolo: Junit ci serve per testare i nostri metodi che implementiamo nelle classi del progetto.

Installazione: digitare sul browser **JUNIT,** e cliccare sul primo link. Per cercarlo cliccate CTRL+F e cercate download and install. Per i più pigri ecco il link: <https://github.com/junit-team/junit4/wiki/Download-and-Install>

Sotto il titolo grande “**Plain-old JAR**” scaricate junit.jar e hamcrest-core.jar

Vi aprirà una seconda finestra da cui vi consiglio di scegliere il jar da ognuno.

Setup: entrate nell’IDE e create il vostro nuovo progetto

(per i più bravi, inserite i jar appena scaricati, sennò vi si presenterà un’altra occasione per inserirli).

Creiamo un contenitore di test, andiamo nel *src* e creiamo una cartella che chiameremo *tests*.

Adesso la cartella tests è un bellissima cartella qualunque.

Rendiamola speciale andando su: file🡪 project structure , a lato della finestra, a sinistra, c’è **modules**.

Vedrete le varie cartelle del progetto, ma senza paura selezionate la vostra cartella *tests* che avete appena creato e **marchiatela** con il bottone verde **Tests**.

Creazione test: adesso potrete andare in una qualsiasi\* classe del progetto, che volete testare, basta andare dentro alla classe, cliccare sul nome della classe e premere ALT+ENTER -> [create test]

Esempio: public class NoteSpese {🡨ALT+ENTER su NoteSpese in questo caso

Nel primo label: testing library, impostate JUNIT4; se si accende una lampadina che vi dice che non riesce a trovarlo, potete cliccare sul bottone FIX, e potete impostare JUNIT nel caso che lo abbiate ancora importato nel progetto; fatto ciò, sempre nella finestrella, potete selezionare le funzioni/metodi che volete testare. Vedrete che cliccando OK, vi si creerà una classe dentro la cartella *tests.*

\*\*\*Nel caso che abbiate più funzioni, possiamo regolamentare l’ordine di esecuzione di ognuna, attraverso le annotazioni; di base usiamo l’annotazione @*test* per fare un test\*\*\*

Come notate, sopra alla funzione che volete testare, c’è l’annotazione @*test,* indica che andremo a fare un test su quel metodo.

A fianco di @*test* possiamo aggiungere due parametri che possono essere:

1. @*Test*(expected=ParseException.class), che si aspetta che il test si aspetta che lanci una eccezione di parsing.
2. @*Test*(timeout=100), che controlla che non sia troppo lunga l’esecuzione della funzione, in caso che passi troppo tempo (settato attraverso il timeout), richiamerà un fallimento del test (utile per vedere se si creano dei loop nella funzione).

Annotazioni:

possiamo dare un ordine di esecuzione nel caso che abbiamo tanti test. Mettiamo sopra ad ogni test una certa annotazione che ora andiamo a vedere:

@Before: questa annotazione viene eseguita prima di ogni test. Solitamente usata per preparare l’ambiente di test (leggere un file, inizializzare una classe con tutti i dati)

@After: questa annotazione viene eseguita dopo ogni test.

@BeforeClass: questo metodo è eseguito una volta, prima di tutti i test. Usato per alcune attività che può essere ad esempio la connessione a un database. I metodi marchiati con questa annotazione vanno definiti in maniera statica per lavorare con **JUNIT**.

@AfterClass: questo metodo è eseguito una volta, dopo che di tutti i test siano finiti. Usato per vedere se tutto sia terminato in maniera corretta e pulita, per esempio si può annotare sulla chiusura di di un database. I metodi marchiati con questa annotazione vanno definiti in maniera statica per lavorare con **JUNIT**.

@Ignore o @Ignore("sto disabilitando il test perché…"): alcune volte volete temporaneamente disabilitare un test o un gruppo di test. Ci basterà appore @Ignore sopra @Test:

Esempio: @Ignore  
 @Test  
 public void modificaNoteSpese() { ... }

Così non verrà eseguito quel test, tuttavia **JUNIT** riporterà quanti test il numero di test ignorati.

*Annotazioni per PRO*: (http://stackoverflow.com/questions/15760881/list-of-annotations-in-junit)

*Assert*: è il metodo che ti dice la risposta del test (se correct, se failure, se error ) e il più importante nel test. I più usati sono *assertEquals*(valore che mi aspetto, valore che la funzione provata mi da)

*assertTrue*(verifica se il risultato del test sia true)

*assertFalse*(verifica se il risultato del test sia false)