# Testing e JUnit

Non è possibile verificare la correttezza assoluta di un programma, in quanto è impossibile testare tutti i possibili input.

Bisogna, quindi, selezionare un piccolo insieme di input ritenuti significativi con cui sia possibile determinare, con una ragionevole confidenza, la correttezza del programma.

# Identificazione dei test

Vi sono 3 categorie in cui è possibile dividere i test, in base ai casi:

- Casi **normali**: almeno un test per ogni comportamento normale identificato dalla specifica del programma
- Casi **errati/anomali**: almeno un test per le condizioni anomale che devono essere gestite dal programma
- Casi **di frontiera**: almeno un test per i casi limite

# Casi normali

Test per ogni comportamento normale identificato dalla specifica del programma del programma. Ci si aspetta un risultato corretto.

**Esempio**: programma che calcola la radice quadrata di un numero e restituisce il risultato positivo.

radice(4) = 2

# Casi errati o anomali

Test di casi errati o particolari che devono essere gestiti dal programma (esempio: input non validi)

**Esempio**: programma che calcola la radice quadrata di un numero e restituisce il risultato positivo.

radice(-20) valore non valido

# Casi di frontiera

Test con input che sono più soggetti a errore

**Esempio**: programma che calcola la radice quadrata di un numero e restituisce il risultato positivo.

radice(0) = 0

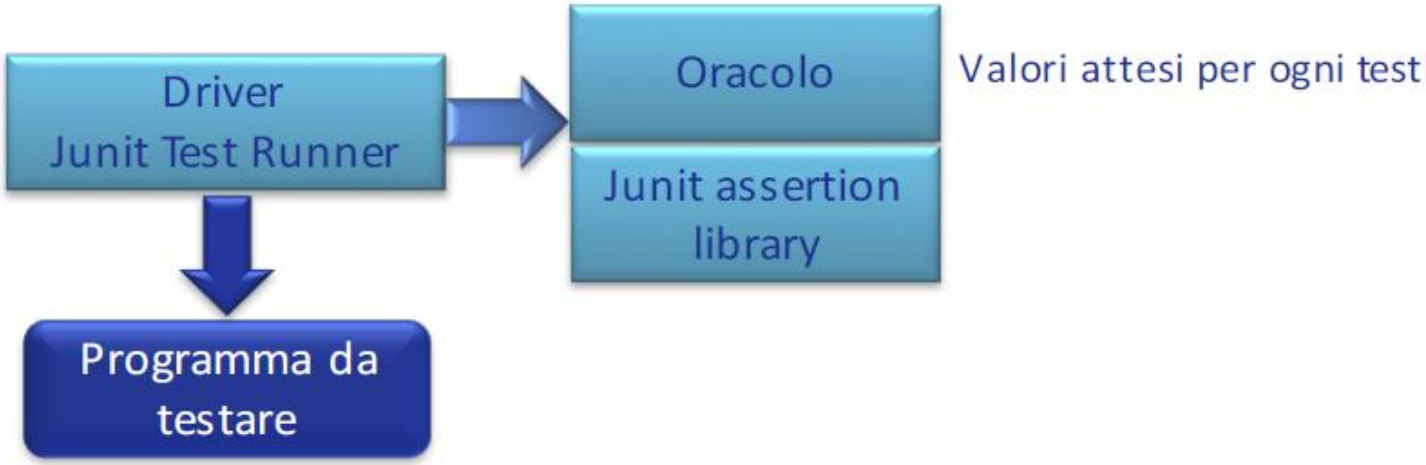
# Esecuzione dei test

I test possono essere implementati nel main del programma, ma ciò risulta:

- Poco flessibile, perché se si vogliono modificare i test bisogna modificare il main
- Poco efficace, perché non è ben chiaro quali prove vengano eseguite e non si possono scegliere quali eseguire

# JUnit

Ambiente di supporto all'esecuzione di test per Java, integrato nella maggior parte degli IDE




**Driver:** sostituisce il main e permette una gestione migliore dei casi di test.

**Oracolo:** specifica i valori attesi per ogni test

# Implementazione in IntelliJ

Creare un nuovo package all'interno del progetto (**test**)

In **Project Structure**, andare in **Modules**, navigare fino al package appena creato e impostarlo come **Test resources** usando il tasto destro.

Selezionare **Libraries** dalla sidebar, cliccare il **+,** scegliere **from Maven** e cercare **junit** all'interno del box.

Selezionare **junit:junit4.1.2** e confermare.

# Implementazione in IntelliJ

Nella riga di dichiarazione della classe, premere **ALT + Invio** e selezionare **Create Test**

Nel popup, scegliere **JUnit 4**, selezionare i metodi per cui si vogliono creare i test e confermare.

Verrà creata una classe Test. Spostarla all'interno del package **test** creato in precedenza.

# Assert

Per determinare la correttezza dei casi di test si utilizza la libreria **assert** di JUnit, composta da diversi metodi:

static void assertEquals(boolean expected, boolean actual)

Asserts that two booleans are equal

static void assertEquals(int expected, int actual)

Asserts that two ints are equal

static void assertEquals(String expected, String actual)

Asserts that two Strings are equal

static void assertFalse(boolean condition)

Asserts that a condition is false

# Assert

static void assertTrue(boolean condition)

Asserts that a condition is true

static void assertNull(java.lang.Object object)

Asserts that an object is null

static void fail()

Fails a test with no message

…