Java SE

- Test del codice con JUnit
  - Supporto di Eclipse al testing
- Logging
  - JUL (java.util.Logger)
  - Cenni su altre librerie
- Progetto di riferimento
  - https://github.com/egalli64/jse *(modulo 4)*

Unit Test

- Verifica (nel folder test) la correttezza di una “unità” di codice, permettendone il rilascio da parte del team di sviluppo con maggior confidenza
- Un unit test, tra l’altro:
  - dimostra che una nuova feature ha il comportamento atteso
  - documenta un cambiamento di funzionalità e verifica che non causi malfunzionamenti in altre parti del codice
  - mostra come funziona il codice corrente
  - tiene sotto controllo il comportamento delle dipendenze

JUnit in Eclipse

- Right click sulla classe (ex: Simple) da testare
  - New, JUnit Test Case
    - JUnit 4 o **5** (**Jupiter**)
    - Source folder dovrebbe essere specifica per i test
  - *Se richiesto,* add JUnit library to the build path
- Il wizard crea una nuova classe (ex: SimpleTest)
  - I metodi che JUnit esegue sono quelli annotati @**Test**
  - Il metodo statico Assertions.fail() indica il fallimento di un test
- Per eseguire un test case: Run as, JUnit Test

Struttura di un test JUnit

- Ogni metodo di test dovrebbe
  - avere un nome significativo
  - essere strutturato in tre fasi
    - Preparazione
    - Esecuzione
    - Assert

```
public int negate(int value) {
    return -value;
}
```

Simple.java

SimpleTest.java

```
@Test
public void negatePositive() {
    Simple simple = new Simple();
    int value = 42;
    int expected = -42;

    int result = simple.negate(value);

    assertThat(result, equalTo(expected));
}
```

@BeforeEach

- I metodi annotati @BeforeEach (Jupiter) o @Before (4) sono usati per la parte comune di inizializzazione dei test
- Ogni @Test è eseguito su una nuova istanza della classe, per assicurare l'indipendenza di ogni test
- Di conseguenza, ogni @Test causa l'esecuzione dei metodi @BeforeEach (o @Before)

```
private Simple simple;

@BeforeEach
public void init() {
    simple = new Simple();
}

@Test
public void negatePositive() {
    int value = 42;

    int result = simple.negate(value);

    assertThat(result, equalTo(-42));
}
```

JUnit assert

- Sono metodi statici definiti in org.junit.jupiter.api.Assertions (Jupiter) o org.junit.Assert (4)
  - assertTrue(condition)
  - assertNull(reference)
  - assertEquals(expected, actual)
  - assertEquals(expected, actual, delta)

```
assertEquals(.87, .29 * 3, .0001);
```

- assert **Hamcrest**-style, usano
  - org.hamcrest.MatcherAssert.assertThat() e matcher (org.hamcrest.CoreMatchers, org.hamcrest.Matchers)

  assertThat(T, Matcher<? super T>) *convenzione opposta ai metodi classici: actual – expected*
  - assertThat(actual, **is**(expected))
  - assertThat(reference, **nullValue**())
  - assertThat(actual, **closeTo**(expected, error))
  - assertThat(actual, **startsWith**("Tom"))
  - assertThat(name, **not**(startsWith("Bob")))

Logging

- Necessità di tener traccia delle attività di una applicazione
- Comunicazione tra chi sviluppa il codice
  - Debugging, analisi dei flussi di esecuzione, …
- L’uso di System.out.println() non è una soluzione accettabile
- Alcune tra le principali librerie utilizzate in Java
  - Java Logging / JUL (java.util.logging)
  - Apache Log4J (noto anche come Log4J 2)
  - SLF4J (+ Logback o altri)

JUL

- Configurazione di default nella JRE/JDK nel folder conf (era in lib fino Java 8): logging.properties
- Ogni programma può avere la sua configurazione
  - Tipicamente gestita come risorsa e messa nel folder src/main/resource (o sub)
  - Lettura via InputStream acceduto da ClassLoader
- Accesso a un logger (singleton) via metodo statico Logger.getLogger()
  - Il parametro è il nome del logger, che dovrebbe essere quello (full) della classe
- Livelli di log tra OFF e ALL – default: attivo da INFO
  - FINEST, FINER, FINE, CONFIG, INFO, WARNING, SEVERE
  - Esempio, per loggare la stringa “message” con severità warning: log.warning(“message”);
- Formatter: prepara la stringa che verrà stampata
- Handler: gestisce la richiesta via console / file
- Per programmi molto semplici: global logger ritornato da Logger.getGlobal()