

JDK vs JRE

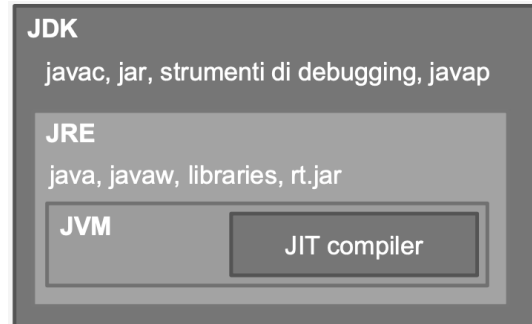
JRE è un'implementazione della **Macchina virtuale Java** ed necessario per l'esecuzione dei programmi (applet ed altre applicazioni) scritti in Java.

Il **Java Runtime Environment** contiene:

- la **Java Virtual Machine**
- le **API** Java standard
- un **launcher** necessario per avviare i programmi già compilati in bytecode.

Il **JDK** è un insieme di software che possono essere utilizzati per sviluppare applicazioni basate su Java.

Il **Java Development Kit** è un **ambiente di sviluppo «a console»**, ovvero non ha un'interfaccia grafica ma le istruzioni si eseguono tramite il prompt dei comandi.



Oltre al JRE ed alla JVM, il JDK contiene diversi strumenti tra cui:

- **javac**: compila il file sorgente in bytecode
- **java**: esegue i file generati dal compilatore
- **javadoc**: serve per creare la documentazione di base del software, sulla base dei commenti inseriti nel codice sorgente
- **jar**: gestisce i file jar (Java Archive, file compressi contenente classi Java)
- **JConsole**: dotato di interfaccia grafica, consente il monitoraggio delle applicazioni Java
- **jdb**: debugger di java a riga di comando

Ciclo di vita del sw: WATERFALL vs AGILE

Waterfall

In questo modello di sviluppo del software, **la sequenza delle fasi del ciclo di vita di un progetto software è strettamente sequenziale e prima di passare alla fase successiva è necessario che quella corrente sia terminata completamente.**



Ciascuna fase, considera il problema da risolvere nella sua interezza.

Limiti

- rischio di aver realizzato un prodotto che soddisfa perfettamente i requisiti inizialmente raccolti che, però, nel frattempo, sono cambiati, o peggio ancora sono stati interpretati male e quindi il software è diverso da quello atteso.

Agile

Nell'approccio Agile **il ciclo di vita è visto come una sequenza di iterazioni in cui ciascuna delle fasi precedenti è applicata ad un sottoinsieme del problema:**

- si raccolgono i requisiti minimi per le funzionalità più importanti
- si progetta il sotto-sistema
- si sviluppa
- si verifica
- si inizia ad usarlo (avvio effettivo in esercizio di un sotto-sistema o componente)
- si ricicla sulle precedenti fasi considerando nuovi requisiti



Mentre nel **WATERFALL** si eseguono tutte le fasi in sequenza e poi si rilascia il SW, nell'approccio **AGILE** il ciclo di vita del sw è visto come una successione di iterazioni (mini cicli, step intermedi, verifiche risultati parziali).

- **Un'iterazione è un arco temporale che va da 2/4 settimane**
- **In ciascuna iterazione prendiamo un pezzo di sw e lo sviluppiamo, testiamo e rilasciamo**
- **Vantaggio:** tra un'iterazione e l'altra ci si accorge subito di eventuali errori ed al termine dello sviluppo dell'iterazione si è certi che quanto sviluppato soddisfa le richieste del cliente.
 - Diminuiscono gli errori relativi a requisiti raccolti sbagliati o vecchi
 - Validazione continua di quel che si sta sviluppando

Agile

L'intero processo di sviluppo è organizzato in «iterazioni» di durata temporale relativamente breve (da 2 a 4 settimane).

- La metodologia prevede una forte e costante interazione tra Committente, Utente e Fornitore, al fine di minimizzare i rischi di fallimento del progetto.

Vantaggi :

- coinvolgimento attivo del committente e degli utenti nel processo di sviluppo;
- aggiornamenti regolari e frequenti sullo stato dell'applicazione;
- validazione continua dei requisiti (dopo ogni iterazione);
- consegna rapida delle funzionalità di base e riduzione del time-to-market;
- pianificazione fissa dei tempi di consegna per funzionalità;
- maggiori test, software migliore.

Agile - SCRUM: sw per il supporto della pianificazione

- offre un boarding plan per pianificare rilasci, test, gestione lavoro etc.

JAVADOC

JavaDoc:

- tool all'interno della JDK messo a disposizione da Java, per realizzare la documentazione tecnica del sw
- è generata in html (così facendo tramite i link, si può navigare facilmente al suo interno)
 - all'interno dei commenti javadoc, si potranno usare i tag html, che saranno correttamente interpretati dal browser

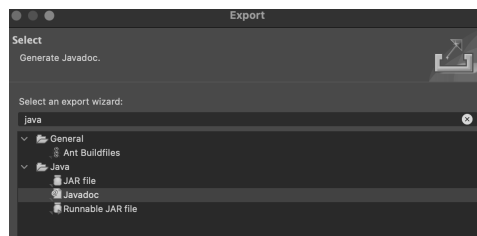
<code>/** <comment> */</code>	Simboli per iniziare un commento javadoc
-------------------------------------	--

LISTA TAG

@author	Identifica il nome dello sviluppatore.
@deprecated	Indica che l'elemento viene ancora mantenuto ma potrà essere eliminato nel successivo rilascio del software. È utilizzato, ad esempio, per indicare che esiste un nuovo metodo e quello contrassegnato con @deprecated è stato mantenuto per garantire la retrocompatibilità. I metodi contrassegnati con questo tag non dovrebbero essere utilizzati, per evitare incompatibilità con versioni di software successive.
@exception	Indica le eccezioni che vengono lanciate da un metodo.
@link	Consente di creare un collegamento ipertestuale ad un link interno alla documentazione o ad una risorsa esterna (tipicamente la pagina di un sito web).
@param	Indica i parametri di un metodo. È necessario inserire un tag per ogni parametro presente nel metodo.
@return	Indica il valore di ritorno di un metodo. Non va usato per costruttori o per metodi che restituiscono <i>void</i> .
@see	Indica un altro metodo o classe a cui far riferimento.
@since	Indica il numero di versione della classe nella quale è stato aggiunto il metodo.
@throws	Indica le eccezioni lanciate da un metodo. (è simile a @exception)
@version	Indica il numero di versione di un elemento (classe o metodo).

GENERAZIONE JAVADOC

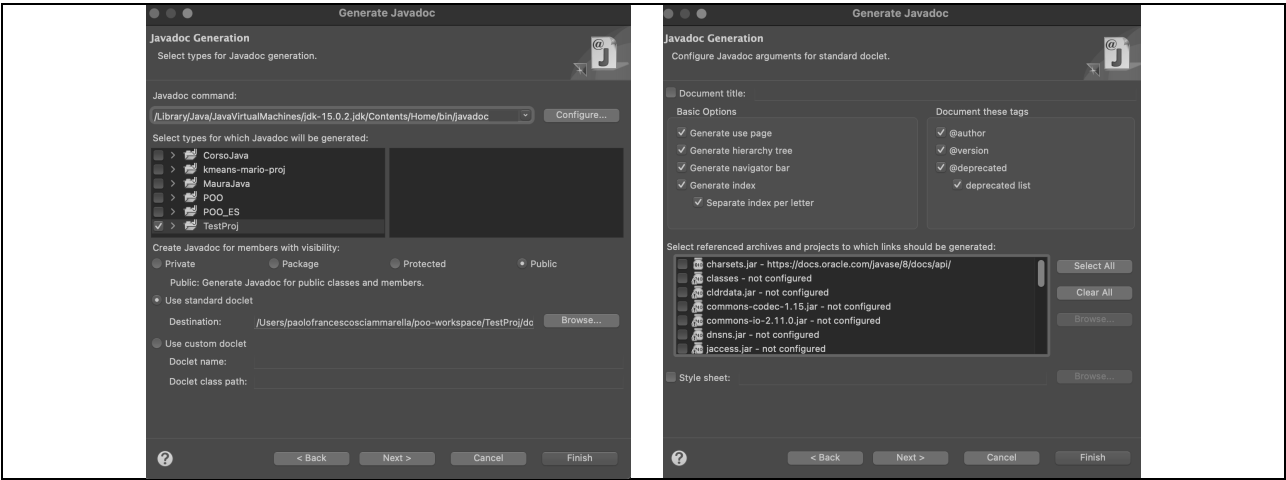
Tasto dx sul progetto – export – Javadoc



Specificare informazioni sul progetto di cui generare il javadoc e della cartella di destinazione.

Allo step successivo, è anche possibile specificare uno stylesheet .css differente per la presentazione della pagina.

La documentazione prodotta, sarà disponibile nella cartella doc: il file contenente le specifiche è index.html



RELAZIONI TRA CLASSI

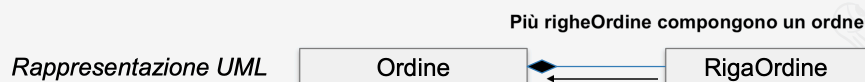
❑ Associazione (Association Relationship)



❑ Aggregazione (Aggregation Relationship)



❑ Composizione (Composition Relationship)



Associazione (Aggregazione e debole)

Si ha una relazione di associazione tra due classi A e B, se da un'istanza di A, è possibile accedere ad un'istanza di B

- Concettualmente: A è associato a B

La relazione di associazione può essere direzionabile: la freccia indica da quale classe (coda) posso accedere alla classe puntata (punta)

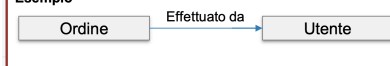
Esempio: da A accedo a B



Codice:

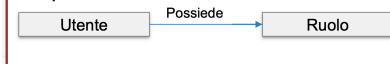
- B parametro di istanza di A

Esempio



La relazione indica che un ordine di acquisto è effettuato da un utente. A partire dall'ordine è possibile risalire all'utente che l'ha creato. Il contrario non si può fare.

Esempio



La relazione indica che un utente ha uno o più ruoli. A partire dall'utente è possibile risalire ai ruoli associati. Il contrario non si può fare.

Esempio



La relazione indica che un conto corrente appartiene ad uno o più utenti. La relazione è bidirezionale poiché a partire dall'utente è possibile risalire al suo conto corrente e viceversa.



Aggregazione

Si ha una relazione di aggregazione tra due classi A e B, se da un'istanza di A, è possibile accedere ad un'istanza di B e **B è una proprietà di A.**

- Una classe A (intero), contiene come parametri una o più classi B (parti), che ne sono logici attributi
- Concettualmente: A è formato/composto da più B

Codice:

- List parametro di istanza di A

	<p>Es: Azienda è un aggregato di Utenti, perché è composta da persone</p> <div> <div> <p>Esempio</p>  <pre> classDiagram Azienda o-- Utente </pre> </div> <div> <p>La relazione indica che un'azienda è composta da persone (rappresentate dalla classe Utente).</p> </div> </div>
Composizione	<p>La relazione di composizione è un tipo più forte di aggregazione ed implica vincoli precisi: le classi che sono parte, non possono esistere senza la classe principale intero.</p> <ul style="list-style-type: none"> • Una classe A (intero), contiene come parametri una o più classi B (parti), che ne sono logici attributi • Concettualmente: A è formato/composto da più B; i B senza A non hanno senso logico (non hanno significato senza che sia istanziata A) • B esiste se e solo se esiste A <p>Codice:</p> <ul style="list-style-type: none"> • List parametro di istanza di A <p>Es: Azienda è un aggregato di Utenti, perché è composta da persone</p> <div> <div> <p>Esempio</p>  <pre> classDiagram Ordine *-- RigaOrdine </pre> </div> <div> <p>Un'istanza della classe RigaOrdine (che contiene il riferimento al prodotto, all'imponibile, etc...) ha senso di esistere solo se esiste un'istanza della classe Ordine</p> </div> </div>