ACADEMY DX R&I – JAVA

UML

Cos’è?

L’UML (Unified Modeling Language) è uno standard per la modellazione del software, utilizzato per specificare visualizzare, costruire e documentare i prodotti del processo di sviluppo. Fornisce agli ingegneri strumenti grafici per rappresentare la struttura e il funzionamento di un sistema in modo standardizzato, facilitando la comunicazione e la progettazione tra architetti e sviluppatori.

Caratteristiche principali

Le principali caratteristiche di UML sono:

* Standardizzato: è gestito dall’OMG – Object Management Gruop, garantendo specifiche uniformi e riconosciute globalmente.
* Visuale: rappresenta concetti e interazioni mediante diagrammi, facilitando la comprensione dei sistemi software.
* Polivalente: applicabile a diversi domini non limitato esclusivamente al software orientato agli oggetti.
* Estensibile: consente personalizzazione attraverso stereotipi, tag value e constraint (vincoli), adattandosi ad esigenze specifiche.

Elementi fondamentali

UML comprende 3 categorie principali di elementi:

* Strutture statiche: rappresentate dai diagrammi delle classi, che descrivono classi, interfacce, collaborazioni e relazioni come ereditarietà e associazioni.
* Comportamenti dinamici: includono i diagrammi di sequenza, di attività, di stato e di casi d'uso, che mostrano interazioni tra oggetti, flussi di lavoro, transizioni di stato e funzionalità viste dagli utenti.
* Aspetti di organizzazione: visualizzati nei diagrammi di pacchetti, di componenti e di distribuzione, utili per gestire i componenti del sistema, le dipendenze e l’allocazione delle risorse fisiche.

Tipi di diagrami UML

……….

JAVA

Storia

Java è un linguaggio di programmazione sviluppato nei primi anni ’90 da un team di ingegneri di Sun Microsystems guidato da James Gosling, con l’obiettivo di creare un linguaggio indipendente dalla piattaforma, riassunto nello slogan WORE – “write once, run anywhere”, ossia “scrivi una volta, esegui ovunque”.

Sebbene la promessa non fosse sempre facile da mantenere già guadagnato popolarità grazie alla sua robustezza scalabilità e affidabilità.

Inizialmente ha cercato di rivoluzionare il web con le applet che però si sono rivelate poco pratiche a causa di problemi di sicurezza e prestazioni. Nonostante ciò, Java si è affermato come uno standard per le applicazioni Enterprise e ha trovato una nuova vita nello sviluppo di app Android.

Nel tempo, Java ha subito numerosi aggiornamenti per rimanere al passo con le esigenze tecnologiche, mantenendosi rilevante anche in un panorama sempre più competitivo. Anche se oggi non è l'unico protagonista della programmazione, il suo ruolo resta fondamentale.

Ciò che contraddistingue Java e la portabilità, data dalla coerenza funzionale. La coerenza funzionale è la capacità del codice OOP di replicare ogni funzione così come è scritta (non da come la interpreta il sistema).

UML e OOP

La Programmazione Orientata agli Oggetti (OOP) e il Linguaggio di Modellazione Unificato (UML) sono strettamente connessi, in quanto entrambi si focalizzano sulla rappresentazione e gestione di sistemi software complessi attraverso il concetto degli oggetti. I principali collegamenti tra OOP e UML sono:

* Astrazione:
  + OOP: consente di concentrarsi su proprietà e comportamenti rilevanti di un oggetto, nascondendo i dettagli implementativi.
  + UML: utilizza i diagrammi di classi per rappresentare visivamente le astrazioni, mostrando attributi e metodi senza dettagli di implementazione.
* Incapsulamento:
  + OOP: è il principio per cui i dettagli interni di un’implementazione di classe sono nascosti all’esterno; quindi, gli oggetti interagiscono tra loro solo attraverso metodi esposti pubblicamente. In sostanza, l’incapsulamento nasconde i dettagli interni di una classe, permettendo l'interazione solo tramite metodi pubblici.
  + UML: rappresenta i livelli di accesso (pubblico, protetto e privato) nei diagrammi di classe.
* Ereditarietà:
  + OOP: consente a una classe di ereditare attributi e metodi da un'altra classe, favorendo il riutilizzo del codice. Permette quindi di creare nuove estrazioni da estrazioni esistenti.
  + UML: mostra l'ereditarietà con frecce vuote che collegano le classi figlie alle classi genitori.
* Polimorfismo:
  + OOP: capacità di un elemento di cambiare forma ma non comportamento o di cambiare comportamento ma non forma. Quindi, permette a oggetti di classe diverse di essere trattati come istanze di una classe comune, purché condividano la stessa interfaccia classe base.
  + UML: rappresenta il polimorfismo tramite diagrammi che evidenziano l'intercambiabilità tra classi che condividono un'interfaccia una gerarchia comune.

Unicità Java

La semantica Java si riferisce al significato delle sue costruzioni sintattiche (variabili, istruzioni, espressioni) che vengono interpretati ed eseguiti dalla macchina. Cinque elementi caratteristici della semantica Java che contribuiscono a definire il suo comportamento:

* Tipizzazione statica e forte: ogni variabile, espressione o blocco di codice ha un tipo definito al momento della compilazione e questi tipi sono rigorosamente controllati. Ciò contribuisce alla sicurezza e alla prevedibilità del codice Java ed evita operazioni non valide tra tipi incompatibili.
* Gestione della Memoria e Garbage Collection: Java gestisce automaticamente la memoria tramite un meccanismo denominato Garbage Collection (GC). Gli oggetti sono allocati nell’heap memory e, quando non sono più raggiungibili attraverso alcun riferimento nel programma, il GC li elimina automaticamente. Ciò reduce il rischio di memory leak e altri problemi di gestione della memoria ma introduce una semantica di performance che gli sviluppatori devono considerare, dato che il timing della GC non è prevedibile.
* Ereditarietà singola e Interfacce: Java supporta l’ereditarietà singola (una classe può ereditare da una sola classe padre). Questo limita la complessità e il rischio di ambiguità. Per supportare la condivisione di comportamenti da più sorgenti Java utilizza le Interfacce (una classe può implementare più interfacce simulando l’ereditarietà multipla).
* Modello di esecuzione Multithreading: permette a più thread in esecuzione di operare contemporaneamente all’interno dello stesso spazio di memoria del processo. Ciò è fondamentale per scrivere applicazioni efficienti e reattive. Tuttavia, bisogna gestire la concorrenza, sincronizzazione e il controllo degli accessi concorrenti alle risorse.
* Eccezioni e gestione degli errori: le eccezioni in Java sono oggetti che rappresebtano condizioni di errore e possono essere “lanciate” o “catturate” usando blocchi try-catch e finally. Cio enatizza la robustezza e la gestione proattiva degli errorri, obbligando gli sviluppatori a gestire esplicitamente le condizioni di errore che possono verificarsi durante l’esecuzione di un programma.

Pratica java

OOP

La programmazione orientata agli oggetti (OOP) è un paradigma di programmazione che permette di definire oggetti software in grado di interagire gli uni con gli altri attraverso lo scambio di messaggi.

Si rivela particolarmente efficace nei contesti in cui è necessario rappresentare le relazioni di interdipendenza tra i concetti da modellare, come contenimento, utilizzo o specializzazione.

Alcuni vantaggi dell’OOP sono:

* Fornisce un supporto naturale alla modellazione software degli oggetti del mondo reale o del modello astratto da riprodurre.
* Permette una più facile gestione manutenzione di progetti di grandi dimensione.
* L'organizzazione del codice sotto forma di classi favorisce la modularità e il riuso del codice.

Alcuni svantaggi dell’OOP sono:

* Necessita di una RAM, e tra l’altro ne ‘mangia’ anche.
* Se non viene composto bene il modello gli oggetti non saranno paritari.
* Se l’ambiente utilizzato non va, non avremo la coerenza funzionale (ad esempio se manca la libreria per ‘leggere’ Java o C).

Nome file e Classe

Le classi definiscono dei tipi di dato e permettono la creazione degli oggetti secondo le caratteristiche definite nella classe stessa. È l'elemento principale dell’OOP.

Grazie alle relazioni di ereditarietà, è possibile creare nuove classi, a partire da quelle esistenti, con caratteristiche aggiuntive.

Una classe è composta da:

* Attributi: variabili e/o costanti che definiscono le caratteristiche o proprietà degli oggetti istanziabili invocando la classe; i valori degli attributi sono inizializzati attraverso il cosidetto costruttore.
* Metodi: procedure che operano sugli attributi;

Piattaforma Java

La piattaforma Java è una ambiente software (API – application programming interface) creato da Sun Microsystems, successivamente acquistata da Oracle Corporation nel 2010. Tale piattaforma fornisce l'ambiente necessario per eseguire programmi scritti in linguaggio Java.

La caratteristica distintiva è la capacità di consentire la creazione e l'esecuzione di applicazioni indipendenti dall'hardware su cui vengono eseguite. Ciò è reso possibile dalla virtualizzazione dell'hardware da parte della piattaforma rendendo il linguaggio Java e i suoi programmi portabili su una vasta gamma di dispositivi e sistemi operativi.

Classe Main e metodo main()

public class Main {

public static void main(String[] args) {

System.out.println("Ciao Mondo ");

}

}

In Java, ogni riga di codice che viene eseguita deve essere contenuta all'interno di una classe. Nel nostro esempio, la classe è chiamata Main. Il nome di una classe deve iniziare sempre con una lettera maiuscola. Inoltre, il nome del file deve corrispondere esattamente al nome della classe. Nel nostro caso il file deve essere salvato come Main.java.

Il metodo main() è sempre richiesto e qualsiasi codice all’interno di esso verrà eseguito.

Metodo System.out.println()

public static void main(String[] args) {

System.out.println("Hello World");

}

Commenti in Java

Variabili

Casting

Scanner

Operatori in Java

Stringhe

Strutture condizionali

Array

ArrayList

Metodi e parametri in Java

Modificatori

Classe ed oggetti

Metodi speciali

Incapsulamento nella pratica

Modificatori di accesso e di non accesso

Get e Set

Ereditarietà nella pratica

Override e Overloading

Polimorfismo nella pratica

Errori ed eccezioni

Classi Wrapper

Inner class

Astrazione nella pratica: abstract e interface