1. Descrivere una classe;

Antonino Caserta

1. Spiegare cosa sono i metodi e spiegare la differenza tra overloading e override;
2. Descrivere il funzionamento del Garbage Collection in Java e spiega l'importanza di questo meccanismo per la gestione della memoria.
3. Spiegare le differenze tra i le variabili (locali, di istanza, di classe, parametri). Spiegare il concetto di istanza e cosa cambia quando si utilizza il modificatore di accesso static.
4. Spiegare i tipi di dati primitivi e il casting

* Qual è il principale scopo di una classe in Java?

A) Implementare algoritmi complessi

**X** B) Fornire un modello per la creazione di oggetti

C) Gestire eccezioni durante l'esecuzione del programma

D) Ottimizzare le performance del codice

* Quale delle seguenti affermazioni è vera riguardo a String e StringBuilder in Java?

**X** A) String è immutabile, mentre StringBuilder è mutabile

B) Entrambe sono immutabili

C) Entrambe sono mutabili

D) String e StringBuilder sono sinonimi

* Qual è il concetto principale dell'ereditarietà in Java?

A) Creare oggetti senza dover dichiarare una classe

**X** B) Permettere a una classe di ereditare le proprietà e i metodi di un'altra classe

C) Limitare l'accesso alle variabili di istanza

D) Creare classi che non possono essere istanziate

* Quando è appropriato utilizzare i getter e i setter in una classe Java?

A) Mai, poiché violano il principio dell'incapsulamento

B) Solo quando si lavora con classi astratte

**X** C) Per fornire un modo controllato per accedere e modificare i dati di una classe

D) Esclusivamente in classi con membri statici

* In che modo StringBuilder è più efficiente di String quando si manipolano grandi quantità di dati?

A) StringBuilder è immutabile

B) StringBuilder è thread-safe

C) StringBuilder utilizza meno memoria grazie alla sua immutabilità

**X** D) StringBuilder è mutabile, consentendo modifiche dirette senza creare nuove istanze

* Quale delle seguenti dichiarazioni è falsa riguardo alla classe Object in Java?

A) Tutte le classi Java ereditano implicitamente dalla classe Object

B) La classe Object contiene i metodi equals() e hashCode()

**X** C) La classe Object è final e non può essere estesa

D) La classe Object fornisce il metodo toString() che può essere sovrascritto

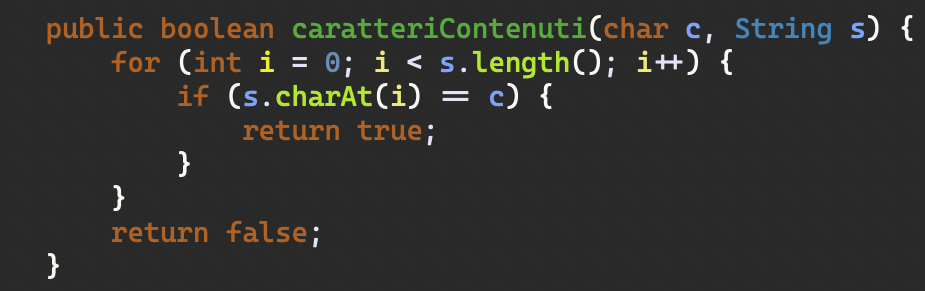
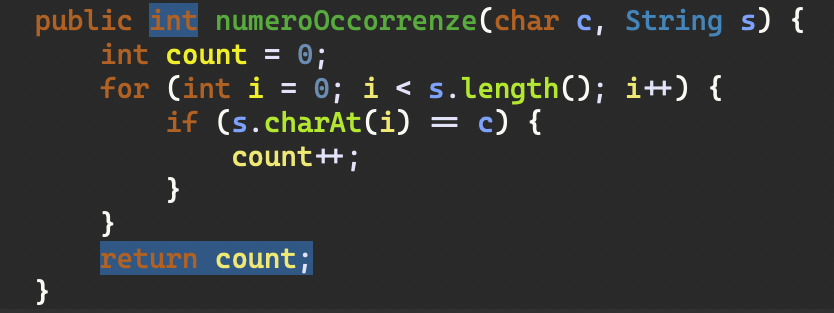
* Qual è il principale vantaggio dell'utilizzo di modificatori di accesso in Java?

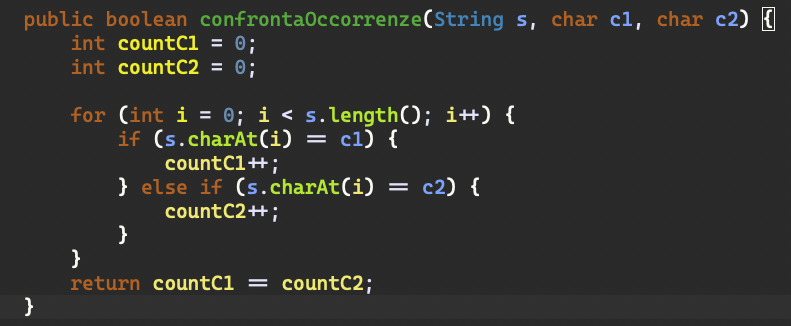
A) Aumentare la complessità del codice

**X** B) Migliorare la leggibilità e la manutenibilità del codice

C) Ridurre le performance del programma

D) Eliminare la necessità di ereditarietà

1. Scrivere un metodo che, dati un carattere c ed una stringa s, restituisce true se c occorre in s, falsealtrimenti.
2. Scrivere un metodo che, dati un carattere c ed una stringa s, restituisce il numero delle occorrenze di c in s.
3. Scrivere un metodo in linguaggio Java che data una stringa s e due caratteri c1 e c2 determini se il numero di occorrenze di c1 in s sia uguale o meno al numero di occorrenze di c2.



RISPOSTE ALLE DOMANDE APERTE:

1. Una classe in Java è un concetto fondamentale nella programmazione orientata agli oggetti, fornendo un modello per creare oggetti e organizzare la logica del programma in unità riutilizzabili. Ogni classe è composta da diversi elementi che definiscono il suo comportamento e le sue caratteristiche. Tra questi elementi, troviamo gli attributi, che rappresentano le proprietà degli oggetti della classe. Gli attributi possono essere variabili di istanza, che hanno un valore specifico per ogni oggetto, o variabili di classe, condivise tra tutti gli oggetti della classe. Questi attributi definiscono lo stato degli oggetti e ne determinano il comportamento. Inoltre, le classi contengono metodi, che definiscono il comportamento degli oggetti e consentono loro di eseguire operazioni specifiche. I metodi possono essere utilizzati per modificare lo stato degli oggetti o per restituire informazioni su di essi. Possono essere definiti come metodi di istanza, che operano su oggetti specifici della classe, e possono essere richiamati su istanze della classe stessa.Un altro elemento importante delle classi è rappresentato dai costruttori, speciali metodi utilizzati per inizializzare gli oggetti quando vengono creati. I costruttori consentono di impostare valori iniziali per gli attributi dell'oggetto o eseguire altre operazioni necessarie prima che l'oggetto sia pronto per l’uso. Infine, l'ereditarietà è un concetto fondamentale che consente a una classe di ereditare attributi e metodi da un'altra classe chiamata superclasse. Questo meccanismo consente di creare gerarchie di classi che condividono funzionalità comuni, riducendo la duplicazione del codice e facilitando la manutenzione del software.In sintesi, una classe in Java rappresenta un modello per creare oggetti e organizzare la logica del programma. Attraverso attributi, metodi, costruttori e l'ereditarietà, è possibile definire il comportamento e le caratteristiche degli oggetti, promuovendo la riutilizzabilità del codice e una migliore organizzazione del software. Esiste un tipo di classe particolare, la classe astratta in Java è una classe che non può essere istanziata direttamente, ma può essere utilizzata come modello per altre classi. Può contenere metodi con implementazioni concrete, così come metodi astratti, che devono essere implementati dalle classi figlie. Una classe astratta può fornire una struttura comune e condivisa per le classi derivate, promuovendo la riusabilità del codice e consentendo una maggiore astrazione.
2. I metodi in Java sono blocchi di codice che eseguono operazioni specifiche su oggetti o tipi di dati. Sono parte integrante delle classi e vengono utilizzati per definire il comportamento degli oggetti. Ogni metodo ha un nome univoco e può avere un insieme di parametri (input), un tipo di ritorno (output) e un corpo, che contiene le istruzioni da eseguire quando il metodo viene chiamato.La differenza principale tra overloading e override riguarda la relazione tra due metodi con lo stesso nome ma in contesti diversi…L'Overloading (sovraccarico) si verifica quando due o più metodi nella stessa classe hanno lo stesso nome ma firme diverse. Le firme dei metodi includono il nome del metodo e il tipo, il numero o la sequenza dei parametri. Il compilatore di Java è in grado di distinguere tra questi metodi in base alle loro firme e di scegliere quale chiamare in base al contesto in cui viene utilizzato il metodo. L'overloading consente di definire più versioni di un metodo con comportamenti diversi, ma con lo stesso nome, per gestire diversi tipi di input o situazioni. L'Override (sovrascrittura) si verifica quando una classe figlia fornisce una nuova implementazione di un metodo già definito nella classe genitore. Per effettuare l'override di un metodo, la firma del metodo nella classe figlia deve essere identica a quella nella classe genitore, compreso il nome del metodo, i tipi dei parametri e il tipo di ritorno. L'override è utilizzato per estendere o personalizzare il comportamento di un metodo ereditato dalla classe genitore nella classe figlia. Quando si chiama un metodo nella classe figlia, verrà eseguita la versione del metodo definita nella classe figlia, anziché quella della classe genitore.
3. La Garbage Collection (GC) in Java è un processo automatizzato essenziale per la gestione dinamica della memoria nell'ambiente di esecuzione Java. La GC funge da custode della memoria heap, la regione di memoria dinamica dove vengono allocate gli oggetti Java durante l'esecuzione del programma.Quando un oggetto Java non è più accessibile dal programma in esecuzione, significa che non ci sono più riferimenti a esso. In questo contesto, la GC interviene identificando e recuperando questi oggetti non più referenziati, al fine di liberare lo spazio di memoria occupato. Nella maggior parte degli ambienti di esecuzione Java, la memoria heap è suddivisa in diverse generazioni, come la "young generation" e la "old generation". La GC utilizza questa suddivisione per ottimizzare le prestazioni. Ad esempio, la "young generation" viene spesso soggetta a un'operazione di GC più frequente, poiché la maggior parte degli oggetti ha una vita breve. Gli oggetti che sopravvivono a diversi cicli di GC nella "young generation" vengono promossi alla "old generation", dove la GC interviene meno frequentemente ma con maggior impatto, poiché gli oggetti qui tendono ad avere una vita più lunga. In conclusione, la Garbage Collection è un componente cruciale dell'ambiente di esecuzione Java, poiché gestisce dinamicamente la memoria, evitando memory leaks e garantendo che lo spazio di memoria venga utilizzato in modo efficiente. La sua attività contribuisce in modo significativo alla stabilità, alle prestazioni e alla sicurezza delle applicazioni Java.
4. Le variabili in Java possono essere classificate in diversi tipi in base al loro ambito di visibilità e al modo in cui sono definite. Ecco le differenze tra variabili locali, di istanza, di classe e parametri:Variabili Locali: Le variabili locali sono definite all'interno di un blocco di codice, come un metodo, un costruttore o un blocco. Esistono solo all'interno del blocco in cui sono dichiarate e vengono distrutte alla fine del blocco. Non hanno un valore predefinito e devono essere inizializzate prima di essere utilizzate. Le variabili di istanza sono dichiarate all'interno di una classe ma all'esterno di qualsiasi metodo. Ogni oggetto istanziato dalla classe ha la sua copia separata di queste variabili. Le variabili di istanza vengono create quando un oggetto è istanziato e vengono distrutte quando l'oggetto è distrutto. Possono avere un valore predefinito o essere inizializzate nei costruttori. Le variabili di classe, anche conosciute come variabili statiche, sono dichiarate utilizzando il modificatore static. Queste variabili sono associate alla classe stessa, anziché a un'istanza specifica della classe. Esiste solo una copia di una variabile di classe, indipendentemente dal numero di istanze create dalla classe. Le variabili di classe vengono inizializzate quando la classe viene caricata e rimangono in memoria per l'intera durata dell'esecuzione del programma. I parametri sono variabili dichiarate all'interno della firma di un metodo o di un costruttore e vengono utilizzati per passare dati a tali metodi o costruttori quando vengono chiamati. Possono essere considerati come variabili locali all'interno del metodo o costruttore e hanno uno scope limitato alla loro dichiarazione. Parlando del concetto di istanza e del modificatore di accesso static… Un’istanza è un oggetto specifico creato a partire da una classe. Ogni volta che si crea un'istanza di una classe utilizzando il costruttore, si ottiene un nuovo oggetto con i suoi dati distinti e le sue variabili di istanza separate Quando si applica il modificatore static a una variabile di istanza, questa diventa una variabile di classe. Ciò significa che tale variabile sarà condivisa da tutte le istanze della classe anziché avere una copia separata per ogni istanza. Quando una variabile di istanza viene dichiarata come static, il suo valore è condiviso da tutte le istanze della classe e può essere accessibile senza creare un'istanza della classe stessa, oltre a ciò cambia il suo comportamento sotto l’aspetto della memoria. Invece di essere associata a ogni singola istanza della classe, la variabile static è allocata in una zona di memoria separata associata direttamente alla classe stessa. Ciò significa che le variabili statiche condividono lo stesso spazio di memoria tra tutte le istanze della classe, piuttosto che avere una copia separata per ciascuna istanza. Questo rende le variabili statiche accessibili da qualsiasi istanza della classe o anche dalla stessa classe senza la necessità di creare un’istanza. Inoltre, le variabili statiche esistono per l’intera durata dell’esecuzione del programma, poiché sono legate alla classe stessa e non alle singole istanze. Anche se non sono presenti istanze della classe in un dato momento, le variabili statiche persistono in memoria.
5. Il tipo int è a 32 bit e può rappresentare numeri interi con segno. Occupa 4 byte di memoria e può memorizzare valori compresi tra -2^31 e 2^31 - 1. È comunemente usato per rappresentare contatori, indici di array e valori numerici generici. Il tipo byte è a 8 bit e può rappresentare numeri interi con segno. Occupa 1 byte di memoria e può memorizzare valori compresi tra -128 e 127. Viene utilizzato in contesti in cui è necessario risparmiare spazio di memoria, ad esempio nella lettura di file binari o nella manipolazione di dati grezzi. Il tipo short è a 16 bit e può rappresentare numeri interi con segno. Occupa 2 byte di memoria e può memorizzare valori compresi tra -2^15 e 2^15 - 1. È meno utilizzato rispetto a int e long, ma può essere utile in situazioni in cui è richiesta una maggiore precisione rispetto a byte ma non si vuole utilizzare l'ampiezza di int. Il tipo long è a 64 bit e può rappresentare numeri interi con segno. Occupa 8 byte di memoria e può memorizzare valori compresi tra -2^63 e 2^63 - 1. Viene utilizzato per rappresentare numeri molto grandi o per garantire precisione nei calcoli aritmetici di grandi dimensioni. I tipi di dato in virgola mobile sono utilizzati per rappresentare numeri con virgola mobile. float è a precisione singola (32 bit), mentre double è a precisione doppia (64 bit). float può rappresentare valori con una precisione approssimativa di 6-7 cifre decimali, mentre double può rappresentare valori con una precisione approssimativa di 15 cifre decimali. Sono utilizzati in situazioni in cui è necessaria la rappresentazione precisa di numeri decimali, come calcoli scientifici o finanziari. Il tipo char è a 16 bit e rappresenta un singolo carattere Unicode. Occupa 2 byte di memoria e può memorizzare caratteri alfanumerici, simboli e caratteri speciali. Viene utilizzato per la manipolazione di stringhe di testo e nella rappresentazione di caratteri in formato Unicode. Il tipo boolean rappresenta i valori booleani true o false. Occupa 1 bit di memoria, ma la dimensione effettiva può variare a seconda dell'implementazione. È utilizzato per valutare condizioni logiche e controllare il flusso di esecuzione del programma. Il casting è un’operazione utilizzata nella programmazione per convertire un tipo di dato in un altro. In Java, il casting può essere implicito o esplicito. Il casting implicito avviene quando un tipo di dato “più piccolo” viene convertito in un tipo di dato “più grande” senza la necessità di specificare esplicitamente il cast. Questo tipo di casting avviene quando il tipo di destinazione ha una maggiore capacità di rappresentazione dei dati rispetto al tipo di origine. Non c’è alcun rischio di perdita di dati durante il casting implicito. D’altra parte, il casting esplicito richiede una specifica esplicita del tipo di dato a cui si desidera convertire. Questo avviene quando un tipo di dato “più grande” viene convertito in un tipo di dato “più piccolo”. Il casting esplicito può comportare la perdita di dati se il valore di origine non può essere completamente rappresentato dal tipo di destinazione.