**EMANUELE BUEMI (test 1)**

1. **Descrivere una classe;**

**Una classe** è un modello o un prototipo per creare oggetti. Al suo interno contiene gli attributi (cioè le proprietà) e i metodi (cioè i comportamenti) che un insieme di oggetti possono avere. Le classi fungono da stampo per creare oggetti specifici, noti come istanze della classe. Le classi in Java sono dichiarate utilizzando il modificatore di accesso pubblic, la parola chiave class (seguita dal nome della classe) e le parentesi graffe {} contenenti le dichiarazioni di variabili di istanza, costruttori e i metodi. È possibile definire una **classe astratta**: essa permette di definire sia metodi astratti (senza implementazione), che devono essere implementati dalle classi figlie concrete, sia metodi non astratti che possono essere ereditati dalle classe figlie. La caratteristica di una classe astratta è che non può essere istanziata per creare un oggetto ma può essere usata solo come superclasse.

1. **Spiegare cosa sono i metodi e spiegare la differenza tra overloading e override;**

I metodi in Java sono blocchi di codice all'interno di una classe che definiscono i comportamenti degli oggetti di quella classe devono avere. Ogni metodo ha un nome univoco che lo identifica e può accettare zero o più parametri come input, eseguire un'azione e restituire un valore.

**L'overloading** dei metodi consente di definire, all'interno della stessa classe, più metodi con lo stesso nome, ma con firme (cioè parametri) diversi. Sarà poi il compilatore Java a scegliere quale metodo chiamare in base al tipo e al numero di argomenti passati. **L'overriding** dei metodi si verifica quando una sottoclasse fornisce una propria implementazione di un metodo che è già stato definito nella classe madre. La sottoclasse si dice che sovrascrive il comportamento del metodo della superclasse.

1. **Descrivere il funzionamento del Garbage Collection in Java e spiega l'importanza di questo meccanismo per la gestione della memoria.**

La **Garbage Collection** è un meccanismo automatico di gestione della memoria in Java che si occupa della liberazione della memoria allocata per gli oggetti che non sono più in uso e che non possono essere raggiunti da nessun riferimento dal programma in esecuzione.

1. **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.**

Le **variabili locali** sono dichiarate all'interno di un metodo ed pertanto esistono solo all'interno di quel blocco di codice in cui sono dichiarate e non possono essere utilizzate al di fuori di esso.

Le **variabili di istanza** sono dichiarate all'interno di una classe ma al di fuori di qualsiasi metodo, costruttore o blocco di codice. Queste variabili vengono allocate nell'heap della memoria e vengono create quando si istanzia un oggetto. Pertanto esistono per tutta la durata dell'oggetto e vengono distrutte quando l'oggetto viene eliminato.

Le **variabili di classe** sono dichiarate con la parola chiave static e appartengono alla classe piuttosto che a una specifica istanza della classe, questo significa che esiste solo una copia di una variabile statica per tutta la classe e tutte le istanze della classe condividono lo stesso valore.

I **parametri** sono variabili passate dentro un metodo e vengono utilizzati per passare dei valori a un metodo quando viene chiamato.

1. **Spiegare i tipi di dati primitivi e il casting**

I **tipi di dati primitivi** sono tipi di dati predefiniti nel linguaggio di Java e non rappresentano degli oggetti, quindi non hanno metodi. Essenzialmente vengono utilizzati per rappresentare valori semplici come numeri interi, numeri in virgola mobile, caratteri e valori booleani.

Il **casting** è il processo di conversione di un tipo di dato in un altro tipo di dato. Possiamo definire due possibili forme di casting: il casting implicito che avviene quando si converte un tipo di dato più piccolo in un altro tipo di dato che ha una dimensione maggiore, e il casting esplicito che avviene quando si converte un tipo di dato più grande in un tipo di dato che ha una dimensione inferiore.

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

A) Implementare algoritmi complessi

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?**

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

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

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

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()

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

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, false altrimenti.**

Immagine che contiene testo, schermata, software, Software multimediale

Descrizione generata automaticamente

1. **Scrivere un metodo che, dati un carattere c ed una stringa s, restituisce il numero delle occorrenze di c in s.**

Immagine che contiene testo, schermata, software, computer

Descrizione generata automaticamente

1. **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.**

Immagine che contiene testo, elettronica, schermata, software

Descrizione generata automaticamente