Domande a pagina 17

1. Quali sono i linguaggi di programmazione da cui hanno preso spunto i ricercatori della Sun Microsystems per lo sviluppo del Java.
   Dai linguaggi di programmazione C e C++.
2. Perché il linguaggio Java è un OOL di tipo "puro"?
   Perché è stato sviluppato appositamente per implementare (realizzare) in un progetto software tutte le caratteristiche della programmazione orientata agli oggetti, e non supporta altri tipi di programmazione come quella procedurale.
3. Quali erano le idee alla base del *Green project* di Sun Microsystems che nel 1991 hanno influenzato lo sviluppo del Java?
   Di creare un linguaggio di programmazione, partendo dal C++, cercando però di modificarlo per renderlo, da un lato, il più semplice possibile e, dall'altro, adatto alla creazione di progetti, anche complessi, in grado di essere eseguiti su macchine differenti.
4. Che cosa si intende per piattaforma, seguendo la terminologia della Sun?
   Un elaboratore individuato da un particolare sistema operativo e dall'hardware, il cui funzionamento è determinato essenzialmente dal tipo di linguaggio macchina proprio di una specifica famiglia di microprocessori.
5. Qual era il nome originario del Java?
   Oak (letteralmente: quercia).
6. In quale caratteristica del linguaggio Java interviene la *Java Virtual Machine*?
   Un programma sorgente Java viene tradotto in un codice denominato bytecode, formato da un insieme di comandi scritti con un linguaggio intermedio tra uno ad alto livello e quello di una macchina ideale (denominata Java Virtual Machine). Il bytecode, svincolato dalle CPU e dai sistemi operativi, viene tradotto dalla Java Virtual Machine nei comandi di un microprocessore reale durante la sua esecuzione.
7. Come è denominato un programma Java scritto per il World Wide Web?
   Applet.
8. Perché il Java è indipendente dalla piattaforma?
   Perché un programma scritto in Java può essere eseguito su una qualsiasi macchina reale in modo indipendente dall'hardware (famiglia di microprocessori) e dal sistema operativo.
9. Come è possibile garantire la sicurezza di programmi Java sviluppati per il Web?
   La sicurezza viene garantita impedendo alle applet di poter svolgere azioni dannose per gli utenti quali, a esempio, la scrittura e la lettura di file nel computer in cui il bytecode dell'applet stesso è eseguito.

Domande a pagina 25

1. Come possono essere classificati i programmi Java?
   In applicazioni stand-alone (che si suddividono a loro volta in applicazioni console e applicazioni con interfaccia utente grafica), applet e applicazioni distribuite (che si suddividono a loro volta in servelet e pagine JSP).
2. Quali sono i principali strumenti software disponibili nell'SDK?
   Il compilatore, l'interprete, il generatore di documentazione e il generatore di file compressi con un insieme di classi.
3. Che cosa rappresenta il bytecode?
   Il bytecode è un'istruzione eseguibile da una macchina virtuale Java (JVM: Java Virtual Machine) programmabile.
4. In che cosa differisce il processo "compilazione più esecuzione" di un programma Java rispetto a uno C/C++?
   Nei linguaggi C/C++, il compilatore traduce un programma sorgente in un codice oggetto scritto con il linguaggio macchina di una specifica famiglia di microprocessori. Il codice oggetto, dopo essere stato allocato nella memoria RAM del computer, viene eseguito da una CPU reale. Nel caso del Java, il compilatore traduce un programma sorgente in un codice "oggetto" scritto con istruzioni bytecode "eseguibili" mediante una macchina virtuale programmabile, che non esiste nella realtà (e che, a secondo della piattaforma, viene definita nel linguaggio macchina proprio di una specifica famiglia di microprocessori). Il compito di trasformare il bytecode nel codice nativo e di eseguirlo in un computer reale è affidato all'interprete Java (contenuto nella Java Virtual Machine). L'azione di un interprete Java è diversa da quella dei software interpreti di altri linguaggi; quest'ultimi traducono in linguaggio macchina direttamente le istruzioni ad alto livello del programma sorgente, mentre un interprete Java traduce in linguaggio macchina dei comandi che sono già una trasformazione del programma sorgente, in una forma più vicina alla macchina, anche se virtuale.
6. Quali devono essere le estensioni dei file con un programma sorgente Java e il bytecode?
   Un programma sorgente java deve avere estensione .java mentre un file contenente il bytecode deve avere estensione .class.
7. Perché un programma sorgente Java è portabile?
   Un programma Java è portabile perché può essere eseguito su qualsiasi piattaforma, in quanto la Sun definisce una JVM per qualsiasi computer (hardware e sistema operativo).
8. Quali sono le fasi principali per l'installazione dell'SDK?
   a. Effettuare il download del file eseguibile con l'ambiente di sviluppo e del file compresso con la documentazione dal sito Web della Sun per il Java all'indirizzo: http://java.sun.com/products
   b. Eseguire il file per il setup dell'SDK e decomprimere il file con la documentazione in una partizione dell'hard disk locale.
   c. Configurare l'SDK.
9. Qual è la funzione principale del JRE?
   L'ambiente Java al tempo di esecuzione (JRE: Java Run time Environment) ha come obiettivo quello di eseguire un codice in bytecode per la Java Virtual Machine in una specifica piattaforma.
10. Qual è la fase che garantisce la sicurezza durante l'esecuzione di un'applet Java?
    Il verificatore del bytecode analizza il codice nella RAM, pronto per essere eseguito, per controllare che il programma non effettui alcuna delle "azioni" proibite definite nelle specifiche della macchina virtuale Java. Nelle applet le regole di sicurezza sono molto più restrittive.
11. Perché l'esecuzione di un codice interpretato è potenzialmente più lenta rispetto all'esecuzione di un programma compilato?
    Perché vi è un passaggio in più tra la compilazione del programma e l'esecuzione dello stesso nel computer. Nel caso del Java, il programma sorgente viene tradotto in un codice bytecode e successivamente questo viene tradotto nel codice oggetto di una specifica famiglia di microprocessori, pronto per essere eseguito da una CPU reale.
12. Quali azioni sono state pensate dai programmatori della Sun per ottimizzare il tempo di esecuzione di un programma in bytecode?
    a. Miglioramento di alcuni costrutti del linguaggio con l'obiettivo di rendere più semplice il bytecode generato e quindi facile da tradurre nel codice macchina degli interpreti run time.
    b. Gestione automatica della memoria da parte della JVM.
    c. Ottimizzazione del codice nativo con cui sono scritte le diverse versioni del JRE.

Domande a pagina 31

1. Come può essere organizzato un generico programma Java?
   Un programma Java è formato da un insieme di una o più classi.
2. Qual è la relazione tra un'applicazione e una classe nel linguaggio Java?
   Un' applicazione viene sviluppata all' interno di una classe, che a sua volta richiama altre classi o istanze di esse.
3. Perché, in un programma Java, la parola chiave CLASS e diversa da quella *class*?
   Perché il linguaggio di programmazione Java è *key sensitive*, ovvero distingue le lettere maiuscole da quelle minuscole.
4. Che cosa rappresenta la sigla JFC (Java Foundation Classes)?
   L' insieme delle classi di base che definiscono la tecnologia Java.
5. Che cosa si intende, al nostro livello di studio, per package?
   Il package è un insieme di classi dedicato alla soluzione di una specifica categoria di problemi.
6. Qual è l'effetto dell'istruzione *import nome_package.**, inserita all'inizio di un codice sorgente Java?
   L' inclusione nel programma di tutte le classi contenute nel package.
7. Qual è il package, automaticamente inserito dal compilatore Java, che fornisce la funzionalità di base del linguaggio in un'applicazione?
   Il package automaticamente inserito dal compilatore java e *java.lang*.
8. Quali sono le caratteristiche di un'applicazione stand-alone console Java?
   Un' applicazione stand-alone console, o semplicemente applicazione console, è eseguita con un' interfaccia di tipo testuale in cui le operazioni di input avvengono mediante la tastiera mentre l' output è visualizzato in forma di testo sullo schermo dell' unità video.
9. Qual è la funzione del metodo main() in un'applicazione console?
   In un' applicazione console, il metodo speciale *main()*, che deve sempre essere presente, dichiara il punto in cui inizia l' esecuzione del programma. All' interno del metodo *main()*, il programmatore deve quindi inserire la sequenza di istruzioni che realizza un algoritmo progettato, richiamando altri metodi della stessa classe e/o elementi (oggetti) di altre classi. Il programma termina dopo l' esecuzione dell' ultima istruzione scritta prima della parentesi graffa chiusa } del metodo *main()*.
10. Che cosa si intende, al nostro livello di studio, per metodo statico di una classe?
    Un metodo statico (o di classe) dichiara un' elaborazione che può essere richiamata anche senza creare nuovi oggetti della classe in cui il metodo stesso è definito.
11. Quali sono i metodi che permettono di effettuare un'operazione di output in un'applicazione console?
    I metodi che permettono di effettuare un' operazione di output sono *print* e *println* appartenenti alla classe **System** (**System.out.***print* e **System.out.***println*).
12. Quali sono i passi e gli strumenti dell'SDK necessari per realizzare un'applicazione console Java?
    a. Traduzione (manuale) del progetto del software nel codice sorgente Java.
    b. Salvataggio del codice sorgente nel file *.java*, con lo stesso nome della classe con il metodo *main()*.
    c. Compilazione del programma sorgente nel bytecode (file *.class*).
    d. Esecuzione del bytecode.
    e. Testing del codice ed eventuale debugging degli errori logici.

Domande a pagina 40

1. Quali sono le caratteristiche di un'applicazione stand-alone con un'interfaccia utente grafica?
   Un'applicazione stand-alone con interfaccia utente grafica (GUI: *Graphical User Interface*), o semplicemente applicazione con GUI, è eseguita in un ambiente in cui l'interazione con l'operatore è basata sull'impiego di una oppure più finestre e di eventi associati al mouse e/o alla tastiera.
2. Quali sono le differenze e le analogie tra un'applicazione stand-alone console e una basata su una GUI?
   La struttura del codice sorgente di un'applicazione stand-alone con GUI è identica a quella di un programma console, con la sostanziale differenza che nel metodo *main()* si devono inserire le istruzioni per la creazione e la gestione dell'interfaccia grafica per l'utente.
3. Quali sono i principali metodi della classe *JOptionPane* che permettono di gestire l'I/O di un'applicazione Java mediante finestre di dialogo?
   I principali metodi della classe **JOptionPane** sono: *setTitle()*, *setSize()*, *setBounds()*, *setDefaultCloseOperation()*, *setVisible()*.
4. Che cosa si intende, in generale, con il termine "applet"?
   Un'applet è un'applicazione Java in bytecode richiamata in una pagina Web ed eseguita dall'interprete Java integrato nel browser di un computer client.
5. Perché l'applet è un esempio di applicazione client-side?
   Perché viene eseguita, mediante il browser, in un computer client.
6. Qual è la funzione dell'attributo CODE e CODEBASE, dei tag <APPLET> e <OBJECT>, durante il richiamo di un'applet in una pagina Web?
   L'attributo CODE specifica il nome del file *.class* dell'applet, mentre CODEBASE definisce l'URL (assoluto o relativo) della cartella del server Web dove sono memorizzati i file dell'applet principale e, eventualmente, di altre classi richiamate.
7. Come viene eseguita un'applet nel browser di un computer client?
   L'interprete Java del browser carica la classe dell'applet (come un nuovo oggetto) e tutte le classi degli oggetti richiamati nel programma, in uno spazio della RAM autonomo e riservato al programma. L'interprete esegue il codice in bytecode dopo aver verificato che non viola le regole di sicurezza per le applet.
8. Quali sono le funzioni dei metodi *init()* e *start()* di un'applet? E le funzioni dei metodi *stop()* e *destroy()*?
   a. Il metodo *init()* alloca nella RAM il codice in bytecode dell'applet, che avviene al primo richiamo della pagina con l'applet.
   b. Il metodo *start()* avvia l'esecuzione dell'applet all'apertura della pagina Web o durante il suo aggiornamento nel browser.
   c. Il metodo *stop()* termina l'esecuzione dell'applet quando l'utente abbandona la pagina Web.
   d. Il metodo *destroy()* dealloca il codice dell'applet dalla memoria centrale del computer. Questa operazione viene eseguita dal JRE in modo automatico alla chiusura del browser.
9. Qual è il metodo di un'applet in cui è conveniente inserire le istruzioni di output sulla finestra grafica del programma nel browser?
   È il metodo *paint()*.
10. Quali sono le fasi principali per la realizzazione di un'applet con l'SDK e il suo inserimento in una pagina Web?
    a. Scrittura del codice sorgente con un editor di testo e salvataggio in un file sorgente *.java*.
    b. Compilazione in bytecode utilizzando il compilatore *javac.exe*.
    c. Inserimento dell'applet in una pagina HTML, impiegando il tag <APPLET> oppure <OBJECT>
    d. Richiamo della pagina web con l'applet mediante un browser oppure lo strumento *appletviewer.exe* dell'SDK.
11. Perché è necessario imporre delle restrizioni per la sicurezza, durante l'esecuzione di un'applet Java?
    Perché l'esecuzione di un'applet permette a un programmatore non autorizzato (hacker o cracker) di poter eseguire un suo programma su un altro computer.
12. Quali sono le principali azioni vietate ai programmatori dai JRE durante l'esecuzione di un'applet?
    a. Aprire una connessione a un server Web diverso da quello in cui è stato effettuato il download dell'applet in svolgimento.
    b. Aprire, leggere, salvare e cancellare file ed eseguire altri programmi, memorizzati nel computer con il browser che interpreta l'applet.

Domande a pagina 46.

1. Perché l'alfabeto del Java è basato sui caratteri Unicode?
   Perché ha il vantaggio di poter codificare i caratteri di tutte le lingue del mondo.
2. Quali sono le regole lessicali per creare un identificatore valido del Java?
   Un identificatore valido del Java può iniziare o con un underscore o con il simbolo del dollaro seguito da una lettera, o deve iniziare con una lettera. Può essere formato da più lettere, cifre o underscore.
3. Come, in genere, i programmatori Java distinguono il nome di una classe da quello di una variabile o di un metodo?
   Il nome di una classe inizia con una lettera maiuscola mentre il nome di una variabile o di un metodo ha la prima lettera minuscola.
4. Perché le parole *class*, *public* o *static* non possono essere nomi validi per una nuova classe?
   Perché sono parole chiave, ovvero, identificatori riservati del linguaggio Java, che indicano al compilatore un'azione, una dichiarazione o un'istruzione eseguibile, che il programmatore vuole realizzare nel programma sorgente.
5. Che cosa rappresenta un letterale nel Java?
   Un letterale è un dato costante (codificato) introdotto da un programmatore in un codice sorgente che rimane inalterato durante la traduzione nel programma eseguibile in linguaggio macchina.
6. Quali sono le principali regole lessicali per scrivere le istruzioni di un programma sorgente Java?
   Nel linguaggio di programmazione Java, il programmatore può scrivere le istruzioni (purché grammaticalmente corrette) con lo stile preferito anche sulla stessa linea.
7. Qual è la funzione dei caratteri di spaziatura in un codice sorgente?
   I caratteri di spaziatura (una sequenza di spazi, una nuova riga, una nuova linea e i tabulatori) sono ignorati dal compilatore ma essenziali per il programmatore, al fine di migliorare la leggibilità del codice prodotto.
8. Come possono essere trasformati due commenti a linea singola in un unico commento multilinea?
   Sostituendo le due doppie barre (//) prima con barra e asterisco (/*) e poi con asterico e barra (*/).
9. Qual è la funzione di un commento di documentazione?
   I commenti di documentazione sono stati introdotti per permettere la generazione automatica di informazioni su un programma in formato HTML (suddivise in un insieme di pagine Web).
10. Come viene generata la documentazione di una classe Java utilizzando il programma javadoc.exe?
    Richiamando il comando con una sintassi del tipo seguente:
    javadoc $[<\textit{-opzioni}>_{\text{repeat}}]_{\text{optional}}$ $[\textit{NomeFile1.java}\ <\textit{NomeFile2.java}>_{\text{repeat}}]_{\text{optional}}$ $[*.\text{java}]_{\text{optional}}$

Domande a pagina 52

1. Su quali elementi è fondato il modello dei dati del linguaggio Java?
   Il modello dei dati del linguaggio Java è basato sui concetti di: variabile, oggetto e tipo di dato. A loro volta i tipi di dato sono suddivisi in: primitivi, classi e interfacce.
2. Che cosa si intende per variabile a livello del codice sorgente e di quello delle memorie?
   Una variabile è un contenitore per i dati elaborati da un programma, al quale il programmatore può associare un identificatore del linguaggio. A livello macchina, le variabili individuano un insieme di celle (o locazioni) della memoria in cui saranno allocati i dati da elaborare. A livello del codice sorgente, il programmatore può associare alle locazioni di memoria un nome simbolico (che soddisfa le regole lessicali del linguaggio) e modificare i dati contenuti nella zona di memoria.
3. Qual è la sintassi generale per dichiarare una nuova variabile di un tipo di dato primitivo?
   Una variabile viene dichiarata indicando il tipo di dato (tipo di dato primitivo o classe) e il suo identificatore (o nome).
4. Come è possibile inizializzare una nuova variabile nella sua dichiarazione?
   Assegnandogli un valore con l'utilizzo dell'operatore di assegnazione =.
5. Come è possibile dichiarare una costante? Quale convenzione usano i programmatori Java per riconoscere una costante in un programma sorgente?
   Una costante viene dichiarata utilizzando la keyword **`final`**. Per una convenzione informale dei programmatori Java, le costanti sono scritte con lettere maiuscole.
6. Come sono classificati i 9 tipi di dato primitivi del Java?
   I tipi di dato primitivi del Java sono classificati secondo il seguente schema:
   a. interi,
   b. reali,
   c. carattere (alfanumerici),
   d. logico,
   e. senza valori.
7. Quali sono le differenze tra un tipo di dato primitivo del Java e un ADT (Abstract Data Type)?
   Un tipo di dato primitivo del Java contiene un numero rappresento in modo approssimato con un numero limita di cifre significative, mentre, in Matematica, una variabile reale (ADT: Abstract Data Type) potrebbe avere un numero illimitato di cifre e quindi una precisione infinita.
8. Quali sono i quattro tipi interi del Java? In che cosa differiscono tra loro?
   I tipi di dato interi del Java sono: **`byte`**, **`short`**, **`int`**, **`long`**. Differiscono tra loro per l'insieme dei valori base che possono assumere.
9. Qual è l'effetto di un owerflow in un'operazione tra due variabili con dati interi?
   La notazione in complemento a due, tipica di questi tipi di dato, è di tipo circolare, per cui se un numero supera ad esempio, il massimo valore positivo rappresentabile, la notazione "ritorna indietro" partendo dal minimo numero negativo. Allo stesso modo, se un numero trabocca il minimo numero rappresentabile, il valore "riparte" da quello maggiore.
10. Come possono essere rappresentati i letterali interi in formato ottale ed esadecimale?
    I letterali interi in formato ottale sono rappresentati facendo precedere il numero da uno *0* (zero). I letterali interi in formato esadecimale sono rappresentati facendo precedere il numero da *0x* o *0X*, inoltre i valori da 10 a 15 sono rappresentati dalle lettere (maiuscole o minuscole) dalla *A* alla *F*.
11. In che cosa si distinguono i due dati reali *float* e *double* del Java?
    Si distinguono per il numero di cifre decimali significative con il quale può essere rappresentato un numero, e per la dimensione del numero.
12. In quali casi un'elaborazione su delle variabili reali può provocare una condizione di owerflow o di underflow?
    Quando il risultato di un' elaborazione supera il valore massimo o il valore minimo rappresentabile.

Domande a pagina 61

1. Quali sono i dati che formano l'insieme base del tipo *char* del Java?
   L'insieme base del tipo **char** è formato dai numeri interi senza segno a 16 bit (da 0 a $2^{-16}-1$), che rappresentano i simboli del codice alfanumerico Unicode.
2. Qual è la funzione dell'operatore + applicato ai caratteri?
3. Quali sono le funzioni delle *sequenze di escape*?
4. Come può essere realizzata una variabile dell'algebra di Boole in un programma Java?
5. Che cosa è il tipo *void*?
6. Che cosa possiamo intendere per un tipo di dato classe? E per un oggetto di una classe?
7. Perché gli oggetti *String* sono trattati dal Java in modo diverso dagli oggetti di altre classi?
8. Qual è l'effetto dell'operatore di concatenamento applicato a oggetti di tipo *String*?
9. Quali sono le differenze tra una stringa della classe *String* e una della classe *StringBuffer*?
10. Quali sono i metodi statici più utili per le classi del Java che corrispondono ai tipi primitivi?
11. Perché in un programma è preferibile usare variabili di tipo primitivo piuttosto che oggetti delle classi *Integer*, *Float*, *Double* ecc.?
12. Come è possibile definire delle elaborazioni matematiche complesse in un programma Java?

Domande a pagina 129

1. Come è possibile elaborare i dai digitati nella riga di comando di un'applicazione console Java?
2. In quali modi e possibile realizzare le operazioni per l'elaborazione dei vettori?
3. Come è definito un array bidimensionale del Java?
   Matrice.
4. Quale ADT permette di realizzare un array a due dimensioni?
5. Come è possibile accedere ai valori memorizzati in un array bidimensionale?
6. Quali differenti valori restituisce l'attributo *length* applicato a un array a due dimensioni?
7. Qual è la sintassi generale per inizializzare un array a due dimensioni?
8. Come è possibile creare un array bidimensionale non rettangolare?
9. Come può essere considerato un array multidimensionale del Java?
10. Perché l'elaborazione di un array multidimensionale può creare dei problemi ai programmatori?
    Perché è necessario gestire più indici e perché occupano ampie zone della memoria.

Domande a pagina 136

1. Che cosa permette di codificare, in generale, il costrutto del Java "metodo" in un programma?
2. Quali sono gli elementi presenti nell'intestazione (o firma) di un metodo?
3. Come si può riconoscere che un metodo non ha alcun parametro di ingresso?
4. Come è formato il corpo di un metodo?
5. Qual è la funzione del comando di salto return in un metodo?
6. Come è eseguito un metodo senza return?
7. Perché l'impiego di più istruzioni return in un metodo può essere svantaggioso?
8. Qual è il tipo di dato di un metodo che non ha istruzioni return nel suo corpo?
9. Qual è la differenza tra gli argomenti e i parametri di un metodo nel Java?
10. Qual è l'effetto sugli argomenti di un metodo nel meccanismo di passaggio per valore?
11. Quali sono le differenze tra il passaggio degli argomenti per riferimento e per valore a un metodo?
12. Quali sono le quattro fasi generali con cui viene eseguita la chiamata di un metodo in un'altra parte del programma?

Domande a pagina 151

1. Che cosa rappresentano i simboli () in una chiamata a un metodo?
2. Quali sono le regole (semantiche) da rispettare nella chiamata di un metodo?
3. Perché un programmatore può considerare un metodo come un "comando" del linguaggio?
4. Qual è il tipo di meccanismo impiegato dal Java nel passaggio di un oggetto a un metodo?
5. Quale tecnica usano i programmatori Java per codificare metodi che restituiscono più di un solo oggetto risultato?
6. Che cosa si intende per overloading dei metodi in una classe Java?
7. In quali casi è possibile applicare l'overloading tra due, oppure più, metodi in una classe?
8. Perché, e in quali casi, l'overloading tra metodi può causare ambiguità? Rispondi alla domanda anche ricorrendo a un esempio concreto?
9. Quali sono le quattro zone in cui è suddivisa la memoria di una JVM?
10. Qual è il meccanismo con cui vengono passati gli argomenti di un metodo in una sua chiamata?
11. Qual è lo schema generale del codice sorgente di un metodo ricorsivo?
12. Quali sono i vantaggi e gli svantaggi di una soluzione ricorsiva rispetto a una iterativa?

Domande a pagina 168

1. Che cosa si intende per astrazione dei dati nella teoria dell'OOP?
2. Quali sono gli elementi principali che compaiono nella dichiarazione di una nuova classe con il costrutto `class` del Java?
3. Com'è organizzato il codice sorgente della classe più semplice a cui possiamo pensare?
4. Perché la classe può essere considerata come la realizzazione di un ADT?
5. Come viene denominato un tipo di dato classe, in modo equivalente, nella tecnologia Java?
   Oggetto.
6. Quali convenzioni adotta il metodo di progettazione OO proposto da Abbott, Booch e Lorensen per evidenziare le parti di un testo candidate a diventare classi, attributi e comportamenti?
   Adotta le convenzioni di evidenziare le parti candidate a diventare: una classe con un bordo, le proprietà (o attributi) con una sottolineatura punteggiata e il comportamento (o elaborazione) con una sottolineatura.
7. Quali possono essere gli elementi (o membri) di una classe del Java?
   Gli elementi (o membri) di una classe possono essere: un attributo, un costruttore o un metodo.

Domande a pagina 174

1. Che cosa si intende per incapsulamento, nella teoria dell'OOP?
   L'incapsulamento è quel meccanismo che permette di includere, all'interno della dichiarazione di una classe, sia il modello dei dati sia gli algoritmi che definiscono l'elaborazione di tutti gli oggetti della classe stessa.
2. Qual è il funzionamento del modificatore di accesso **`private`** scritto prima di un elemento di una classe?
3. Qual è la differenza, ai fini della visibilità, tra un membro dichiarato **`public`** rispetto a un elemento privato?
4. Che cosa realizza un attributo (di istanza) dichiarato in una classe?
5. Qual è l'ambito di un attributo in una classe?
   La variabile è visibile da tutti gli altri elementi di una classe, ma se non viene dichiarata come statica, per poter essere utilizzata è necessario creare una copia (o istanza) della classe stessa denominata "oggetto".
6. Come avviene l'accesso agli attributi pubblici di una classe in un programma sorgente?
   Utilizzando l'operatore punto con una sintassi del tipo: *nomeOggetto.nomeAttributoPubblico*.
7. Come devono essere dichiarati gli attributi di una classe per realizzare in modo corretto il meccanismo dell'incapsulamento dell'OOP?
   Gli attributi di una classe devono essere dichiarati come privati, utilizzando il modificatore per il controllo dell'accesso **`private`**.
8. Qual è la funzione di un attributo di classe (o statico)? Come può essere dichiarato nel corpo di una classe?
9. Quali sono i modi con cui può essere inizializzata una variabile di classe?
   Assegnando direttamente un valore nella fase di dichiarazione della variabile stessa; oppure attraverso un blocco statico, ovvero un blocco di elaborazione eseguito una sola volta durante la fase di allocazione del bytecode della classe in memoria, e che assegna alle variabili dei valori.
10. Che cosa realizza un metodo dichiarato in una classe?
11. Qual è la funzione di un metodo pubblico in una classe?
12. Qual è la differenza tra un metodo privato rispetto a uno pubblico, in una classe?
13. Che cosa rappresenta un messaggio secondo la terminologia dell'OOP?
14. Qual è il compito svolto da un metodo di classe? Come può essere dichiarato nel corpo di una classe?
15. Qual è la funzione dei metodi pubblici per l'accesso agli attributi privati? Quali sono gli scheletri dei codici sorgente per un metodo di accesso in lettura e per uno in scrittura?
    Questi metodi servono per leggere o scrivere il valore degli attributi (o variabili) di istanza. Gli scheletri dei codici sorgente per i metodi di accesso in lettura o scrittura agli attributi di istanza sono i seguenti:

```
public void scriviAttributo(tipoDatoAttributoPrivato nomeParametro) {
        nomeAttributoPrivato = nomeParametro;
}
public tipoDatoAttributoPrivato leggiAttributo() {
        return nomeAttributoPrivato;
}
```

Domande a pagina 182

1. Quali sono i compiti svolti da un costruttore? Quali sono le differenze nella dichiarazione di un costruttore rispetto a un metodo della classe?
2. In quale punto del codice sorgente il programmatore richiama un costruttore?
   Quando vuole creare un oggetto (copia o istanza) di una classe.
3. Che cos'è il costruttore di default? Come agisce se è applicato a un nuovo oggetto?
4. Quali sono le linee guida generali per la dichiarazione di una nuova classe?
5. Che cosa rappresenta l'interfaccia di una classe?
   Rappresenta la parte pubblica della classe visibile agli utenti esterni. L' interfaccia coincide con le intestazioni (firme) dei metodi pubblici.
6. Quale deve essere la visione di una classe per gli utenti (altri programmatori)?
7. Che cosa si intende con la sigla API?
   Application Programming Interface (interfaccia di programmazione delle applicazioni); ovvero la documentazione di tutte le interfacce del Java.
8. Quali elementi deve contenere e come può essere prodotta la documentazione di una classe?
9. Quali sono i vantaggi dell'applicazione del meccanismo dell'incapsulamento in una classe?
   - ✓ La protezione degli attributi privati dai tentativi di accesso errati degli utenti. (altri programmatori).
   - ✓ L' occultamento degli algoritmi che implementano l' elaborazione, la cui modifica da parte dell' utente potrebbe comportare bug logici.
   - ✓ La possibilità di modificare gli algoritmi che realizzano l' elaborazione senza modificare l' interfaccia, ma fornendo soltanto una nuova versione del file .class in bytecode.
   - ✓ La riusabilità (reusability) degli oggetti in software creati per obiettivi anche molto differenti da quelli originali per cui è nata una classe.
10. Come sono realizzati nel Java la dichiarazione e l'implementazione di una classe dell'OOP?

Domande a pagina 188

1. Che cos'è un oggetto? Come può essere denominato un oggetto in modo alternativo?
2. Che cosa contiene una variabile reference? Come può essere dichiarata in un programma?
3. Qual è l'ambito di una variabile reference?
4. Che cosa rappresenta la parola chiave `null`?
5. Quali sono le operazioni compiute dall'operatore `new`?
6. Che cosa contiene la memoria *heap*?
7. Quali sono le fasi del ciclo di vita di un oggetto?
8. Che cosa si intende per garbage collection? Quali sono le sue funzioni svolte per la gestione degli oggetti?
9. Quali sono i vantaggi del garbage collection per i programmatori?
10. Che cosa rappresenta il metodo *finalize()*?

Domande a pagina 206

1. Che cosa rappresenta la parola chiave **`this`** del linguaggio? Quali sono le sue principali applicazioni?
2. Qual è la funzione principale dell'operatore **`instanceof`**?
   L'operatore **`instanceof`** serve per verificare se un determinato oggetto è una istanza di una determinata classe o se può essere convertito in quella classe.
3. Quali sono le caratteristiche del passaggio di un oggetto come parametro di un metodo?
4. Com'è possibile realizzare, in un programma Java, l'ADT record logico?
   Creando una classe, i cui attributi privati sono i campi descritti nel tracciato del record stesso.
5. Com'è possibile tradurre l'ADT tabella in un modello object oriented?
   Utilizzando un array monodimensionale (vettore) di variabili reference di un tipo di classe (record).
6. Che cosa rappresenta la classe *Object* per le altre classi del Java?
7. Com'è possibile scrivere un metodo generale per visualizzare lo stato interno incapsulato degli oggetti di una classe?
8. Com'è possibile annidare un oggetto di una classe nella dichiarazione di un'altra?
9. Com'è possibile dichiarare una classe interna a un'altra? Quali sono le caratteristiche di una classe interna.
   Dichiarando una nuova classe nel corpo di una classe già esistente. Una classe interna a accesso a tutti i membri (attributi di istanza e metodi), anche privati, della classe in cui è contenuta.

Domande a pagina 219

1. Che cosa rappresenta un modulo object oriented nel Java?
2. Quali sono le principali linee guida per costruire software sorgenti efficienti, riusabili, portabili e ben documentati?
3. Com'è organizzato un programma sorgente object oriented multifile?
4. Quali sono le principali linee guida per il testing di un software multifile?
5. Che cosa rappresenta un'interfaccia nel Java?
6. Quali sono le regole che deve seguire una classe che implementa un'interfaccia?
7. Una classe può implementare più di una sola interfaccia?
   Si.
8. Quali sono le convenzioni grafiche che si impiegano nei modelli object oriented per rappresentare che una classe implementa un'interfaccia?
   Si utilizza una freccia tratteggiata orientata da una classe verso un tipo di dato interfaccia.
9. Quali sono le relazioni tra le variabili reference, gli oggetti e le interfacce?
10. Quali sono le principali applicazioni delle interfacce?
11. Com'è possibile produrre la documentazione di un'interfaccia?
12. Quali sono le principali differenze e analogie tra una classe e un'interfaccia?

Domande a pagina 228

1. Quali sono i package alla base della grafica del Java 2?
2. Qual è la differenza tra un contenitore e un componente nella grafica del Java?
3. Quali sono le principali classi che permettono di creare una finestra nel sistema grafico del Java?
4. Che cosa rappresenta un evento nella grafica del Java?
5. Come viene modellato un evento, ad esempio del mouse oppure della tastiera?
6. Qual è la funzione di un'interfaccia per l'ascolto di eventi? Che cosa è un oggetto ascoltatore?
7. Che cosa contiene un'interfaccia per l'ascolto di eventi?
8. Che cosa rappresenta un oggetto sorgente di eventi?
9. Qual è il compito svolto da un metodo di registrazione applicato a un oggetto sorgente di eventi?
10. Perché il sistema per la gestione degli eventi del Java è denominato a delega degli eventi?
11. Quali sono i principali passi operativi che un programmatore deve seguire per poter elaborare degli eventi generati in un determinato programma?
12. Qual è la funzione delle classi adattatore e di quelle interne nella gestione degli eventi del Java?
13. Che cosa rappresenta un pannello? Qual è la sua principale funzione?
14. Che cosa si intende per layout di un pannello? E per gestore di layout?