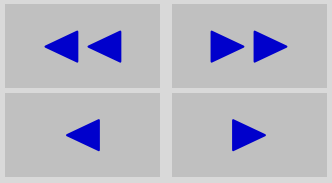
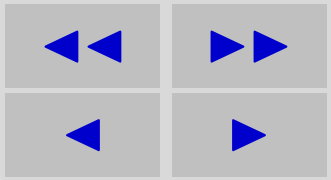
Capitolo 3

Domande vero-falso

Selezionando una risposta, il bottone corrispondente diventa verde se la risposta è giusta, altrimenti diventa rosso.



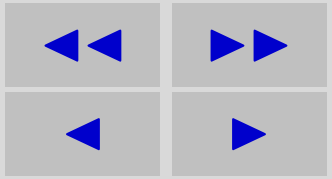

P.G.MARRANO



1. Tutti gli operatori in Java hanno associatività da sinistra verso destra.

- ☐ Falso
- ☐ Vero

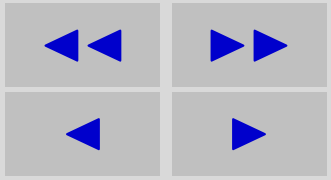
☐



2. L'ordine in cui un programma esegue le istruzioni è detto precedenza.

- ☐ Falso
- ☐ Vero

☐



3. Ci sono diverse istruzioni Java che consentono ai programmatori di specificare che l'esecuzione deve saltare ad una determinata istruzione: in questo caso, si parla di precedenza.

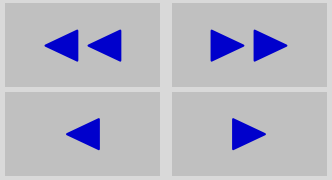
- ☐ Falso
- ☐ Vero

☐

4. La ricerca di Nicklaus Wirth ha dimostrato che i programmi possono essere scritti senza istruzioni **goto**.

- ☐ Falso
- ☐ Vero

☐



5. La struttura `switch` è detta una struttura di selezione multipla perché seleziona tra molte diverse azioni.

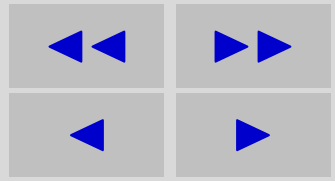
- ☐ Falso
- ☐ Vero

☐

6. Il blocco di istruzioni all'interno di una struttura `if` consiste di un'espressione il cui valore di ritorno può essere `true` oppure `false`.

- ☐ Falso
- ☐ Vero

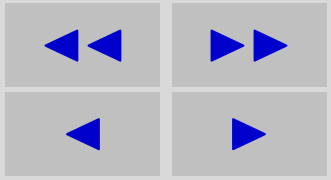
☐



7. La struttura di selezione `if-else` permette al programmatore di specificare che una diversa azione deve essere eseguita quando la condizione è vera da quando la condizione è falsa.

- ☐ Falso
- ☐ Vero

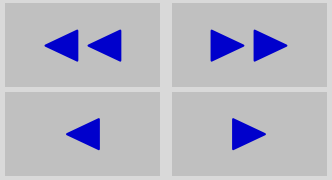
☐



8. L'operatore condizionale è l'unico operatore ternario di Java: esso riceve due operandi.

- ☐ Falso
- ☐ Vero

☐



9. Inserire un punto e virgola dopo la condizione causa un errore logico nel caso di una struttura `if` ed un errore sintattico nel caso di una struttura `if-else` (se la parte `if` ha un corpo non vuoto).

- ☐ Falso
- ☐ Vero

☐

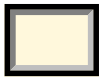
10. Un blocco non può contenere dichiarazioni di variabili.

- ☐ Falso
- ☐ Vero

☐

11. La precedenza di `++` è minore di quella di `<=`.

- ☐ Falso
- ☐ Vero






12. Talvolta, un algoritmo contiene decisioni nelle quali una variabile oppure un'espressione viene separatamente confrontata con ciascuno dei valori integrali (ovvero, valori del tipo `byte`, `short`, `int`, `long` e `char`) che essa può assumere, e diverse azioni vengono eseguite. Java fornisce la struttura `case` di selezione multipla per gestire questo tipo di decisioni.

- ☐ Falso
- ☐ Vero

☐

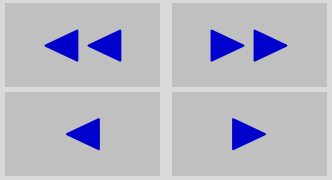
13. I casi non esplicitamente verificati in un'istruzione `switch` senza un caso `default` sono ignorati.

- ☐ Falso
- ☐ Vero

14. La struttura `switch` consiste di una serie di etichette `case` e di un caso obbligatorio `default`.

- ☐ Falso
- ☐ Vero

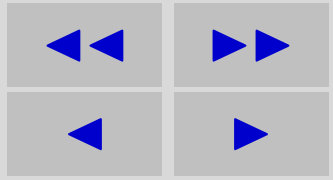
☐



15. Se si elencano, una di seguito all'altra, le etichette di casi diversi (ad esempio, `case` 1: `case` 2: senza istruzioni in mezzo) allora lo stesso insieme di azioni verrà eseguito per ciascuno dei casi.

- ☐ Falso
- ☐ Vero

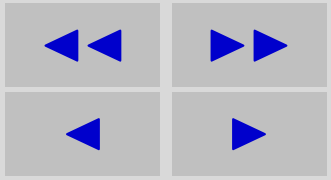
☐



16. Java fornisce operatori logici che possono essere usati per formare condizioni più complesse. Gli operatori logici sono `&&`, `&`, `||`, `|`, `^` e `!`.

- ☐ Falso
- ☐ Vero

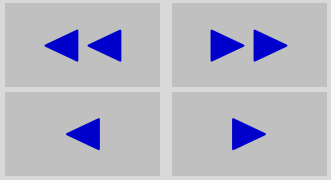
☐



17. L'operatore `&&` ha la stessa precedenza dell'operatore `||`.

- ☐ Falso
- ☐ Vero

☐



18. Java fornisce l'operatore `!` per consentire al programmatore di invertire il significato di una condizione.

- ☐ Falso
- ☐ Vero

☐

◀◀ ▶▶
◀ ▶

19. A differenza degli operatori logici `&&`, `&`, `||`, `|` e `^` che combinano due condizioni (operatori binari), l'operatore di negazione logica ha una sola condizione come operando (operatore unario).

- ☐ Falso
- ☐ Vero

20. La seguente è una valida espressione Booleana in Java: `(1 < count < 10)`.

- ☐ Falso
- ☐ Vero

21. Il valore dell'espressione di controllo di una struttura `switch` può essere di uno qualunque dei tipi di dato primitivi.

- ☐ Falso
- ☐ Vero

☐

22. Gli operatori di confronto (come `<` e `!=`) hanno una precedenza maggiore degli operatori `&&` e `||`.

- ☐ Falso
- ☐ Vero

23. È possibile che il corpo di un ciclo `while` venga eseguito zero volte.

- ☐ Falso
- ☐ Vero

24. Una struttura di ripetizione consente al programmatore di specificare che un'azione sia ripetuta fintanto che una certa condizione rimane vera.

- ☐ Falso
- ☐ Vero

☐

25. Se la condizione di una struttura `while` è inizialmente vera, le istruzioni del corpo non verranno mai eseguite.

- ☐ Falso
- ☐ Vero

26. La ripetizione controllata da un contatore utilizza una costante, detta contatore, per controllare il numero di volte che un insieme di istruzioni verrà eseguito.

- ☐ Falso
- ☐ Vero

☐

27. La ripetizione controllata da un contatore richiede il nome di una variabile di controllo (detta contatore), il valore iniziale del contatore, l'incremento (o decremento) con il quale il contatore è modificato ad ogni iterazione del ciclo, e la condizione che verifica il valore finale del contatore (ovvero, se il ciclo deve proseguire).

- ☐ Falso
- ☐ Vero

☐

28. Poiché i numeri in virgola mobile possono essere approssimati, il controllo di un ciclo che faccia uso di variabili di tipo reale può causare valore imprecisi e, quindi, verifiche inaccurate di terminazione.

- ☐ Falso
- ☐ Vero

☐

29. La struttura di ripetizione `for` consente di gestire tutti i dettagli di una ripetizione controllata da un contatore.

- ☐ Falso
- ☐ Vero

30. Supponiamo che un programma cicli 10 volte facendo uso della condizione di continuazione `counter<=10`. Se il programmatore ha erroneamente scritto `counter<10`, il ciclo sarebbe eseguito solo nove volte. Questo è un frequente errore sintattico.

- ☐ Falso
- ☐ Vero

☐

31. Quando una struttura `for` inizia la sua esecuzione, è certo che il suo corpo sarà eseguito almeno una volta.

- ☐ Falso
- ☐ Vero

☐

32. Non è possibile modificare il valore di una variabile di controllo nel corpo di una struttura `for`.

- ☐ Falso
- ☐ Vero

33. La ripetizione controllata da una sentinella è spesso detta ripetizione definita in quanto il numero di ripetizioni è noto prima che il ciclo inizi la sua esecuzione.

- ☐ Falso
- ☐ Vero

☐

34. La seguente struttura `for` (di cui solo l'intestazione è mostrata) modifica la variabile di controllo da 100 ad 1 con un decremento di 1:

```
for (int i = 100; i<=1; i--)
```

- ☐ Falso
- ☐ Vero

☐

35. Nella struttura `while`, la condizione di continuazione del ciclo viene verificata all'inizio del ciclo prima che il corpo del ciclo sia eseguito.

- ☐ Falso
- ☐ Vero

☐

36. Quando un ciclo `do-while` termina, l'esecuzione continua con l'istruzione successiva alla riga `do`.

- ☐ Falso
- ☐ Vero

37. L'istruzione `break`, quando eseguita in una struttura `while`, `for`, `do-while` oppure `switch`, causa l'uscita immediata da quella struttura. L'esecuzione continua con la prima istruzione successiva alla struttura.

- ☐ Falso
- ☐ Vero

38. L'istruzione `continue`, quando eseguita in una struttura `while`, `for` oppure `do-while`, salta le rimanenti istruzioni del corpo di quella struttura e procede con la successiva iterazione della struttura.

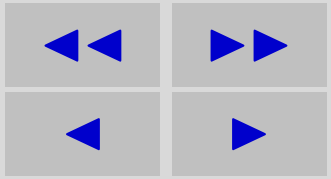
- ☐ Falso
- ☐ Vero

☐

39. L'istruzione `break` può interrompere solo la struttura `while`, `for`, `do-while` oppure `switch` che la racchiude direttamente.

- ☐ Falso
- ☐ Vero

☐



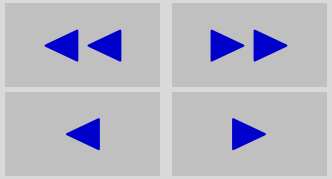
40. I metodi non di tipo `void` devono contenere una o più istruzioni `return`.

- ☐ Falso
- ☐ Vero

41. È possibile che il corpo di un ciclo `do-while` venga eseguito zero volte.

- ☐ Falso
- ☐ Vero

☐



42. I seguenti due frammenti di codice:

```
int i = 10, sq = 0;
do {
  sq += i * i;
  i++;
} while (i <= 10);
```

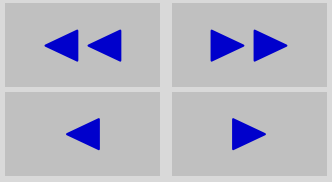
e

```
int i = 10, sq = 0;
while (i <= 10) {
  sq += i * i;
  i++;
}
```

sono equivalenti (dopo la loro esecuzione le variabili `i` e `sq` hanno gli stessi valori)

☐ Vero

☐ Falso



43. Consideriamo i seguenti due frammenti di codice:

```
int x = -1; if(x<0) x=1; else x=2;
int x = -1; if(x<0) x=1; if (x>=0) x=2;
```

Al termine della loro esecuzione, la variabile `x` avrà lo stesso valore.

- ☐ Vero
- ☐ Falso

☐