Prova d'Esame per il Corso di Metodologie di Programmazione

Università degli Studi di Roma "La Sapienza"

19 Maggio 2021

Durante l'esame non è consentito l'utilizzo di alcunché. Non è consentito inoltre l'utilizzo della matita o di penne il cui colore sia diverso dal nero o dal blu. Il ritiro dalla prova equivale al mancato superamento dell'esame stesso.

Nome e Cognome:

Matricola:

1. Descrivere i modificatori di visibilità utilizzati all'interno delle classi.

   - **public**: i membri di una classe dichiarati public possono essere accessibili sia all'interno che all'esterno della classe;
   - **private**: i membri di una classe dichiarati private sono visibili solamente all'interno della classe in cui sono stati dichiarati;
   - **protected**: i membri di una classe dichiarati protected sono visibili all'interno della classe in cui sono stati dichiarati e in tutte le sottoclassi che la ereditano.

     I membri di una classe dichiarati senza alcun modificatore di visibilità ne ottengono uno di *default*, tramite il quale possono essere manipolati da ogni classe del pacchetto a cui appartengono.

2. Spiegare il concetto di interfaccia ed il suo utilizzo. Produrre un esempio (con codice minimale) che comprenda la definizione di una interfaccia insieme ad un metodo che la preveda come parametro per svolgere un compito a scelta.

   Un'interfaccia è un tipo di dato astratto utilizzato per definire il comportamento di una o più classi. Esse vengono dichiarate tramite la parola chiave *interface* ed implementate da una classe tramite la parola chiave *implements*. Una classe può implementare una o più interfacce, diversificandosi rispetto al meccanismo dell'ereditarietà in cui una classe può ereditare al più da una superclasse.
   In termini strutturali, un'interfaccia non possiede variabili d'istanza ma solamente variabili di classe. Inoltre, tutti i suoi metodi sono astratti e con modificatore di visibilità *public*. Infine, un'interfaccia non prevede costruttori.

   Un esempio di codice minimale è riportato all'interna della cartella *es_2*.

3. Quante iterazioni eseguono i seguenti costrutti iterativi, supponendo che la variabile contatore i non venga modificata all'interno di ciascun ciclo?

   (a) `for(int i = 1; i <= 10; i++){...}`

   (b) `for(int i = 0; i < 10; i++){...}`

   (c) `for(int i = 10; i > 0; i--){...}`

   (d) `for(int i = -10; i >= 0; i++){...}`

(e) `for(int i = 10; i >= 0; i++){...}`

(f) `for(int i = -10; i <= 10; i = i + 2){...}`

(g) `for(int i = -10; i <= 10; i = i + 3){...}`

- a): 10
- b): 10
- c): 10
- d): 0
- e): non termina
- f): 11
- g): 7

Il codice di tale esercizio è disponibile all'interno della cartella *es_3*.

4. Scrivere un programma che generi una sequenza di 20 lanci casuali di un dado, memorizzando i risultati in un array. Successivamente, visualizzare i valori ottenuti, identificando soltanto la ripetizione più lunga e racchiudendola tra parentesi tonde come in questo esempio:

   1 2 5 5 3 1 2 4 3 (2 2 2 2) 3 6 5 5 6 3 3 3 3 1

   In caso di più ripetizioni dalla stessa lunghezza massima, contrassegnare solamente la prima.

   La soluzione di tale esercizio è disponibile all'interno della cartella *es_4*.

5. Il responsabile amministrativo di un albergo registra le transazioni in un file di testo. Ogni riga contiene le seguenti informazioni, separate da "punti e virgola":

   - il nome del cliente;
   - il servizio venduto (ad esempio, cena, conferenza, alloggio, etc.);
   - l'importo pagato;
   - la data dell'evento.

   Scrivere un programma che legga un tale file di testo e visualizzi l'importo totale relativo a ciascun servizio, segnalando un errore se il file non esiste oppure se il suo formato non è corretto. A seguire viene riportato un esempio del contenuto del file:

   Mario Bianchi;Cena;29.95;6/7/2020
   Giovanna Rossi;Conferenza;499.00;8/9/2020
   Manuela Verdi;Alloggio;23.45;10/10/2020
   Giulia Gialli;Cena;93.00;11/12/2020
   Michele Apicella;Cinema;10.00;1/1/1970

   La soluzione di tale esercizio è disponibile all'interno della cartella *es_5*.

6. Un aeroporto ha un'unica pista. Quando la pista è occupata, gli aerei che vogliono atterrare o decollare devono attendere. Realizzare un programma di simulazione usando due code, una per gli aerei in attesa di decollare e una per quelli in attesa di atterrare. Gli aerei in attesa di atterrare hanno la precedenza. L'utente del simulatore può digitare i seguenti comandi: **takeoff** codiceDelVolo, **land** codiceDelVolo, **next** e **quit**. I primi due comandi inseriscono il volo nella coda corrispondente (decollo per **takeoff** e atterraggio per **land**). Il comando **next** pone termine all'operazione in corso (decollo o atterraggio) e fa partire la successiva, visualizzando l'azione da intraprendere (decollo o atterraggio) e il codice del volo. Il comando **quit**, ovviamente, termina la simulazione.

   La soluzione di tale esercizio è disponibile all'interno della cartella *es_6*.