

DOMANDE

Programmare in Java

- 1 Quali di queste affermazioni, riguardanti le caratteristiche principali del linguaggio Java, sono vere (V) e quali false (F)?
 - a) È un linguaggio orientato agli oggetti
 - b) Permette di realizzare applicazioni portabili
 - c) Genera programmi solo per la piattaforma Linux
 - d) Gestisce automaticamente la memoria (allocazione e deallocazione)

- 2 Quali di queste affermazioni, riguardanti il *bytecode*, sono vere (V) e quali false (F)?
 - a) Non è un codice portabile
 - b) È il risultato della compilazione di un programma Java
 - c) Può essere eseguito direttamente
 - d) Deve essere interpretato per essere eseguito
 - e) È contenuto nei file con estensione .class

- 3 Attraverso quale modalità operativa i programmi scritti in Java sono portabili, cioè eseguibili senza modifiche su piattaforme diverse ?
 - a) interpretazione
 - b) editing
 - c) compilazione
 - d) debugging

- 4 Qual è il corretto significato dell'acronimo API ?
 - a) Active Processing Interface
 - b) Active Programming Institute
 - c) Application Processing Institute
 - d) Application Programming Interface

- 5 Quali di queste dichiarazioni corrisponde alla corretta dichiarazione del metodo *main*?
 - a) public static void main (String args)
 - b) public void main (String args[])
 - c) public static void main (String args[1])
 - d) public static void main ()

- 6 Qual è l'estensione di un file sorgente di Java?
 - a) .class
 - b) .txt
 - c) .cpp
 - d) .java

- 7 Associa ad ogni operazione della colonna a sinistra il corrispondente programma o comando scegliendo nella colonna a destra.

a) editare il codice Java	1) javac
b) compilare il codice Java	2) java
c) eseguire il programma Java	3) Blocco note

V F
V F
V F
V F

V F
V F
V F
V F
V F

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

Elementi base del linguaggio

- 8 Per ognuno dei nomi elencati, indicare se è un identificatore valido in Java (si/no).
 - a) eta media
 - b) 1peso
 - c) COLORE_SFONDO
 - d) Velocita-max
 - e) NumeroAbbonati
 - f) Città

- 9 Qual è la forma della dichiarazione di una variabile con assegnamento di un valore iniziale?
 - a) <tipo> <nome variabile> = <valore iniziale>;
 - b) <nome variabile> <tipo> = <valore iniziale>;
 - c) <nome variabile> = <valore iniziale> <tipo>;
 - d) <tipo> <valore iniziale> = <nome variabile>;

- 10 Qual è la forma della dichiarazione di una costante di tipo carattere ?
 - a) final char RISPOSTA_POSITIVA = Y;
 - b) char final RISPOSTA_POSITIVA = 'Y';
 - c) final int RISPOSTA_POSITIVA = 'Y';
 - d) final char RISPOSTA_POSITIVA = 'Y';

- 11 Quale tra i seguenti tipi di dato messi a disposizione da Java non fa parte dei tipi di dato primitivi ?
 - a) numerici
 - b) carattere
 - c) array
 - d) booleani

- 12 Associa a ogni tipo di dato numerico a sinistra la corrispondente occupazione di memoria:

a) float	1) 8 bit
b) int	2) 16 bit
c) byte	3) 32 bit
d) double	4) 32 bit
e) long	5) 64 bit
f) short	6) 64 bit

- 13 Quali di queste affermazioni, riferite alla conversione tra tipi, sono vere (V) e quali false (F)?
 - a) La conversione dal tipo *float* al tipo *int* viene eseguita automaticamente da Java V F
 - b) La conversione dal tipo *float* al tipo *int* comporta una perdita di informazione V F
 - c) La conversione dal tipo *float* al tipo *int* è chiamata promozione V F
 - d) Dividendo un numero di tipo *int* per un *int* si ottiene un numero di tipo *float* V F
 - e) La conversione dal tipo *int* al tipo *float* viene eseguita con il casting V F

- 14 Quali di queste affermazioni, riferite al concetto di *casting* in Java, sono vere (V) e quali false (F)?
 - a) È sempre eseguito automaticamente V F
 - b) In alcuni casi deve essere il programmatore a richiederlo V F
 - c) È uno strumento per convertire un tipo di dato in un altro V F
 - d) La compilazione individua eventuali errori di casting V F
 - e) Il casting dal tipo *double* al tipo *int* è automatico V F

15 Supponendo che la variabile *a* di tipo intero assuma ogni volta il valore iniziale 5, quale valore assume alla fine di ogni gruppo di istruzioni?

- a) $a += 4;$ b) $b = a * 2;$ c) $b = 18 \% a;$
 $b = a - 8;$ $a -= b;$ $a *= 2;$
 $a = 10 / a;$ $a = 3;$ $a = a + b;$
-

16 Indica il valore assunto dalla variabile *risultato* al termine di ogni gruppo di operazioni.

- a) $\text{risultato} = 0;$ b) $\text{risultato} = 0;$
 $\text{num} = 5;$ $\text{num} = 5;$
 $\text{risultato} = (\text{num} -) + 3;$ $\text{risultato} = (- - \text{num}) + 3;$
-

Strutture di controllo

17 Quali di queste affermazioni, riferite alle strutture di controllo Java, sono vere (V) e quali false (F)?

- a) `if (...) {} else {}` è una struttura di selezione
b) `for (...) {}` è una struttura di selezione
c) `while (...) {}` è una struttura di iterazione
d) `switch (...) {case: ...break;}` è una struttura di iterazione
e) `try {...} catch (...)` è una struttura di iterazione

V F
V F
V F
V F
V F

18 Come vengono tradotti in Java gli operatori logici AND, OR e NOT?

Operatore logico	Operatore in Java
AND	
OR	
NOT	

19 Supponendo che la variabile *x* di tipo intero assuma ogni volta il valore iniziale 3, quale valore assume alla fine dei tre segmenti di codice?

- a) $\text{if } (x > 3)$
 $\{$
 $\quad x *= 2;$
 $\}$
 else
 $\{$
 $\quad x += 2;$
 $\}$
- b) $\text{if } (x == 3)$
 $\{$
 $\quad x = 5;$
 $\}$
 $\text{if } (x <= 5)$
 $\{$
 $\quad x -= 2;$
 $\}$
- c) $\text{if } (x >= 3)$
 $\{$
 $\quad x++;$
 $\quad \text{if } (x > 4)$
 $\quad \{$
 $\quad \quad x = 10;$
 $\quad \}$

20 Quale valore assume la variabile *y* alla fine del segmento di programma?

```
int y=0;
int a=2;
while (a <= 1024)
{
    y++;
    a *= 2;
}
```

21 Quale valore assume la variabile *y* alla fine del segmento di programma?

```
int y=1;
boolean fine;
do
{
    y++;
    fine = (y > 10);
}
while (!fine);
```

22 Quale valore assume la variabile *y* alla fine del segmento di programma?

```
y=30;
for (int i=0; i <= y; i=i+2)
{
    y -= 5;
```

Array

23 Metti in ordine, indicando un valore da 1 a 3, i passaggi che si devono seguire per la creazione di un array.

Passaggi per la creazione	Ordine
Inizializzazione	
Dichiarazione	
Allocazione	

24 Qual tra le seguenti dichiarazioni è corretta per allocare un array di 10 elementi?

- a) `valori[10] = new double[];`
b) `valori[] = new double[10];`
c) `valori = new double[10];`
d) `valori = double[10];`

25 Qual è l'errore contenuto nella dichiarazione di un array: `int lista[50];`?

- a) Doveva essere `int lista[50] = new int[];`
b) L'array non è stato inizializzato
c) Non si può indicare la dimensione dell'array nella dichiarazione
d) Non si può dichiarare un array di tipo int

26 Completa le frasi seguenti utilizzando una tra le parole elencate alla fine della domanda.

- a) Si può indicare la dimensione dell'array usando la parola
b) Per accedere ad un elemento dell'array si indica l'indice tra parentesi
c) Se si usa un indice maggiore della dimensione dell'array viene generata l'eccezione
d) Un array dichiarato ma non ancora allocato assume il valore speciale
graffe, tonde, dimens, ArrayIndexOutOfBoundsException, length, ArrayLimitException, null, int, quadre, void

27 Qual tra le seguenti dichiarazioni è corretta per assegnare all'ultimo elemento di una matrice 4x4 il valore 99?

- a) `matrice[matrice.length] = 99;`
b) `matrice[0][0] = 99;`
c) `matrice[3][3] = 99;`
d) `matrice[4][4] = 99;`

Eccezioni

- 28 Quali di queste affermazioni, riferite al concetto di eccezione, sono vere (V) e quali false (F)?
- È una situazione anomala che può verificarsi durante l'esecuzione
 - Si verifica in fase di compilazione
 - Se non gestita causa la terminazione del programma
 - È controllata con il blocco try ... catch ...

V	F
V	F
V	F
V	F

PROBLEMI**Programmare in Java**

- Effettuare un collegamento al sito <http://www.oracle.com/technetwork/java/index.html> per individuare la versione più recente del JDK (Java Development Kit). Accedere alla sezione Java SE (Standard Edition), fare il download del JDK ed eseguire l'installazione sul computer.
- Scrivere, usando Blocco Note, il seguente programma che calcola la somma di due numeri e salvarlo con il nome appropriato.

```
class Somma
{
    public static void main(String args[])
    {
        int a = 3;
        int b = 8;
        int somma = a + b;
        System.out.println(somma);
    }
}
```

- Compilare ed eseguire il programma precedente.

Elementi base del linguaggio

- Scrivere un programma contenente la seguente dichiarazione di una variabile intera e verificare quale errore viene generato in fase di compilazione.
int volume = 15.0;
- Scrivere un programma che dichiara una costante intera *perciva* il cui valore è 20. Successivamente modificare il valore di *perciva* assegnando il valore 21. Verificare che, compilando il programma, viene visualizzato il messaggio di errore "cannot assign a value to final variable *perciva*".
- Scrivere un programma che dichiara una variabile di nome *lunghezza* e di tipo byte. Assegnare alla variabile il valore 140. Verificare che, compilando il programma, viene visualizzato il messaggio di errore "possible loss of precision".
- Calcolare quanti byte occupano le seguenti variabili:
 - 5 variabili di tipo float
 - 7 variabili di tipo short
 - 4 variabili di tipo double
 - 4 variabili di tipo long
 - 10 variabili di tipo int
 - 10 variabili di tipo char

Input e output

- Scrivere un programma che restituisce il seguente output a video:
TITOLO:
"La divina commedia"
AUTORE
D. Alighieri
 - Scrivere un programma che legge da tastiera due valori di tipo *double* e esegue la divisione restituendo il risultato sullo schermo.
 - Scrivere un programma che legge da tastiera il nome e il cognome di una persona e stampa sul video il nome completo.
 - Scrivere un programma che legge da tastiera un numero e comunica con un messaggio se il numero è pari o dispari. Si ricorda che un numero è pari se il resto della divisione per 2 restituisce 0.
 - Scrivere un programma che legge da tastiera la misura del lato e calcola l'area e il perimetro di un quadrato.
 - Dato il prezzo di un prodotto e la percentuale di sconto, calcolare e visualizzare il prezzo scontato.
- *
- Strutture di controllo**
- Avendo la lunghezza di un'auto espressa con un valore in millimetri, costruire un programma per convertire il valore nel formato metri, centimetri tralasciando gli eventuali millimetri residui.
 - Supponendo che una rete televisiva abbia trasmesso 5.876 minuti di documentari in un anno, scrivere un programma per convertire il valore nel formato giorni, ore, minuti.
 - Avendo la variabile intera *h* e usando gli operatori, scrivere le condizioni booleane che siano vere quando:
 - *h* si trova nell'intervallo [2, 7],
 - *h* si trova nell'intervallo [-4, 3],
 - *h* è uguale a 0 oppure è maggiore di 100,
 - *h* non è negativo,
 - *h* è diverso da 100.
 - Riscrivere la seguente struttura *while* usando la struttura iterativa *for*.


```
int c;
c = 8;
while (c > 0)
{
    c -= 2;
}
```
 - Calcolare quante volte viene eseguito il ciclo precedente.
 - Scrivere un programma che stampa i primi 100 numeri pari usando la struttura di ripetizione *while*.

- 20 Scrivere un programma che stampa i primi 100 numeri pari usando la struttura di ripetizione *do while*.
- 21 Scrivere un programma che stampa i primi 100 numeri pari usando la struttura di ripetizione *for*.
- 22 Scrivere un programma che produce a video la tavola pitagorica.
- 23 Dato un elenco di 10 prodotti con descrizione e prezzo, visualizzare il prezzo massimo.
- 24 Indicare qual è il comportamento del seguente frammento di codice

```
int i;
for (i=1; i <=10; i = i - 1)
{
    System.out.println(i);
}
```

- 25 Verificare se il seguente frammento di codice rappresenta un ciclo infinito.

```
int i = 5;
while (i < 300)
{
    if (i < 100)
    {
        i++;
        break;
    }
    i--;
    continue;
}
```

Array

- 26 Dichiarare e allocare un array contenente 12 numeri a doppia precisione.
 - 27 A partire dalla seguente definizione di array,
- ```
int tab[][] = new int[7][4];
```
- scrivere due frammenti di codice per calcolare il totale di ogni riga e il totale di ogni colonna della matrice.
- 28 Scrivere un programma per definire un array di 5 numeri e per valorizzarli con 5 numeri casuali tra 1 e 90. Visualizzare successivamente i numeri scelti.
  - 29 Dopo aver acquisito da tastiera un array di 10 numeri a singola precisione, sommare le sue componenti e visualizzare il risultato.
  - 30 Dopo aver acquisito da tastiera due matrici 2x2 di numeri interi, calcolare la somma delle due matrici sommando le singole componenti.
  - 31 Definire un array di dimensione 10 e successivamente provare ad accedere all'elemento di posizione 20 gestendo l'eccezione che viene generata.
- parte seconda**  
La programmazione
- capitolo 3**  
Le basi del linguaggio Java
- ## PROBLEMI DI RIEPILOGO
- La realizzazione dei seguenti programmi deve essere preceduta da un'analisi che soddisfi i seguenti punti:*
- descrizione generale del problema
  - dati di input e di output
  - pseudocodifica
  - programma Java.
- 32 Dati due numeri interi compresi tra 0 e 49, generati casualmente, visualizzare il numero più grande.
  - 33 Generare cinque numeri casuali, reali e compresi tra 0 e 1, e calcolarne la media.
  - 34 Una scuola è composta da N classi. Per ogni classe, viene inserito da tastiera il numero di studenti. Calcolare quanti studenti frequentano la scuola e in media quanti studenti ci sono per classe.
  - 35 Scrivere un programma che ricevendo da linea di comando la base e l'altezza di un rettangolo, restituisce il valore dell'area.
  - 36 Dato come input il tempo in ore, minuti e secondi, convertire in secondi. Per esempio 5 ore, 9 minuti, 24 secondi = 18564 secondi.
  - 37 Dopo aver ottenuto da tastiera un numero decimale, calcolare la rappresentazione dello stesso numero nel sistema binario.
  - 38 Scrivere un programma che continua a richiedere numeri finché viene inserito il valore zero. Alla fine indica quanti sono stati i numeri positivi e i numeri negativi inseriti.
  - 39 Un anno è bisestile se è divisibile per 4 e non è divisibile per 100. Sono però bisestili anche gli anni divisibili per 400. Scrivere un programma che, inserendo un anno da tastiera, risponda se è un anno bisestile o no.
  - 40 Un'urna contiene venti palline numerate da 1 a 20. Creare un programma che simuli un'estrazione casuale dall'urna e che consideri come vittoria l'estrazione di un numero pari.
  - 41 Data la velocità espressa in Km/h calcolare la conversione in m/s.
  - 42 Nella trasmissione di dati tramite reti di calcolatori, l'unità di misura utilizzata è il bit. Ricevuto in input un numero indicante i bit trasmessi, calcolare a quanti byte, Kb, Mb, Gb corrisponde. Per esempio 84.054.321 bit = 10.506.790 byte = 10.260,5 Kb = 10,02 Mb = 0,0098 Gb.
  - 43 Scrivere un programma che calcola il prodotto tra due matrici A e B di dimensioni rispettivamente 3x2 e 2x3.
  - 44 Problema di ricerca in un array: dopo aver generato un array di 30 numeri casuali tra 0 e 99, determinare se è presente il numero 50.
- 148
- 149

**PROGRAMMA JAVA** (*ProgAnalisi.java*)

```
import java.io.*;

class ProgAnalisi
{
 public static void main(String argv[])
 {
 InputStreamReader input = new InputStreamReader(System.in);
 BufferedReader tastiera = new BufferedReader(input);

 String frase = "";
 GestoreFrasi gestore = new GestoreFrasi();

 System.out.println("Inserisci il testo da analizzare:");
 try
 {
 frase = tastiera.readLine();
 }
 catch(IOException e) {}

 gestore.analizza(frase);

 System.out.println("\nNumero di caratteri = " + frase.length());
 System.out.println("\nVOCALI");
 System.out.println("Numero = " + gestore.vocali);
 System.out.println("Frequenza = " + gestore.getFreqVocali());

 System.out.println("\nSPAZI");
 System.out.println("Numero = " + gestore.spazi);
 System.out.println("Frequenza = " + gestore.getFreqSpazi());
 }
}
```

**AUTOVERIFICA**

Domande da 24 a 26 pag. 228  
Problemi da 35 a 39 pag. 231



**MATERIALI ONLINE**

**8. Documentazione delle librerie Java**

**DOMANDE**

**Programmazione ad oggetti**

- 1 Qual è il significato di OOP?
  - a) Object Oriented Paradigm
  - b) Object Operational Programming
  - c) Object Oriented Programming
  - d) Object Operational Paradigm
- 2 Quali di queste affermazioni, riferite al concetto di oggetto, sono vere (V) e quali false (F)?
  - a) Le caratteristiche di un oggetto descrivono il suo stato V F
  - b) I comportamenti di un oggetto descrivono il suo stato V F
  - c) I comportamenti si riferiscono alle funzionalità dell'oggetto V F
  - d) Un oggetto si può descrivere graficamente con un diagramma degli oggetti V F
  - e) L'oggetto è alla base della programmazione strutturata V F
- 3 Come si può rappresentare la definizione di una classe?
  - a) Elencando solo i suoi attributi
  - b) Elencando solo i suoi metodi
  - c) Elencando sia gli attributi che i metodi
  - d) Facendo alcuni esempi
- 4 Come viene rappresentata graficamente una classe?
  - a) Con il diagramma degli oggetti
  - b) Con il diagramma delle classi
  - c) Con il grafo delle classi
  - d) Con il grafo di gerarchia

**Dichiarazione delle classi e uso degli oggetti**

- 5 Quali di queste affermazioni, riferite alle classi, sono vere (V) e quali false (F)?
  - a) In ogni classe ci deve essere un metodo chiamato *main* V F
  - b) La dichiarazione di una classe inizia con la parola chiave *class* V F
  - c) Ogni classe viene memorizzata in un file con estensione *.class* V F
  - d) Le classi contengono solo la dichiarazione degli attributi V F
- 6 A che cosa serve l'operatore *new*?
  - a) A creare una classe
  - b) A creare un attributo
  - c) A creare un metodo
  - d) A creare un oggetto
- 7 Quale tra le seguenti rappresenta la corretta dichiarazione di un attributo?
  - a) tipo livelloDiVisibilità nomeAttributo;
  - b) livelloDiVisibilità tipo nomeAttributo;
  - c) livelloDiVisibilità nomeAttributo tipo;
  - d) nomeAttributo livelloDiVisibilità tipo;

- 8 Per ognuno dei seguenti elementi presenti nella dichiarazione di un metodo, indica se è obbligatorio oppure facoltativo.

|                           | <b>Obbligatorio</b> | <b>Facoltativo</b> |
|---------------------------|---------------------|--------------------|
| Nome                      |                     |                    |
| Blocco di istruzioni      |                     |                    |
| Tipo di valore di ritorno |                     |                    |
| Elenco di parametri       |                     |                    |
| Livello di visibilità     |                     |                    |

- 9 Qual è la caratteristica del metodo a, la cui intestazione è *public void a()*?
- Non restituisce nessun valore
  - Non contiene istruzioni
  - Può essere visto solo all'interno della classe in cui viene dichiarato
  - È un metodo statico

- 10 Quale tra queste istruzioni viene usata dai metodi per restituire un valore?
- void
  - return
  - this
  - System.exit(1)

- 11 In quale altro modo si possono chiamare le variabili dichiarate all'interno dei metodi?
- Attributi
  - Variabili di istanza
  - Variabili locali
  - Variabili globali

- 12 Quali di queste affermazioni, riferite ai metodi costruttori, sono vere (V) e quali false (F)?
- Vengono eseguiti automaticamente quando si crea un nuovo oggetto
  - Restituiscono un valore di tipo int
  - Solitamente contengono le istruzioni di inizializzazione
  - Hanno lo stesso nome della classe a cui appartengono
  - In ogni classe deve essere dichiarato un costruttore

V F  
V F  
V F  
V F  
V F

- 13 Qual è il meccanismo che fa interagire tra loro gli oggetti?
- Lo scambio di attributi
  - Lo scambio di messaggi
  - L'ereditarietà
  - L'incapsulamento

V F  
V F  
V F  
V F

- 14 Quali di queste affermazioni, riferite ai riferimenti nulli, sono vere (V) e quali false (F)?
- Il riferimento null rappresenta un oggetto dichiarato ma non allocato
  - Non è possibile assegnare esplicitamente il valore null
  - Accedendo a un attributo di un oggetto null si genera l'eccezione *NullPointerException*
  - Il riferimento null è un valore numerico intero

### Mascheramento dell'informazione e interazione tra oggetti

- 15 Quali di queste affermazioni, riferite al concetto di *information hiding*, sono vere (V) e quali false (F)?
- Viene realizzato in pratica dall'interfaccia dell'oggetto
  - Viene realizzato in pratica dalla sezione privata dell'oggetto
  - Corrisponde al concetto di encapsulamento
  - Maschera la struttura dell'oggetto
  - Maschera l'oggetto rendendolo inutilizzabile

V F  
V F  
V F  
V F  
V F

- 16 Completa le frasi seguenti utilizzando una tra le parole elencate alla fine della domanda.
- Il metodo ..... è usato per leggere il valore di un attributo.
  - Il metodo ..... è usato per modificare il valore di un attributo.
  - L'elenco dei metodi pubblici indica l' ..... di una classe.
  - La sezione ..... comprende gli attributi e i metodi non accessibili dall'esterno.  
*interfaccia, istanza, get, protected, pubblica, set, privata, hiding, costruttore*

### Ereditarietà e polimorfismo

- 17 Qual è la corretta dichiarazione per specificare che la classe A è sottoclasse della classe B?
- class A
  - class B extends A
  - class A extends B
  - class A extend B

V F  
V F  
V F  
V F

- 18 Quali di queste affermazioni, riferite all'ereditarietà in Java, sono vere (V) e quali false (F)?
- Ogni classe può avere una sola sopraclasse (vale l'ereditarietà singola)
  - La sottoclasse si dichiara con la parola chiave extends
  - Gli oggetti creati dalle sottoclassi possono accedere solo agli attributi delle sopraclasse e non ai metodi
  - Gli attributi private di una classe sono ereditabili
  - Il riferimento speciale super serve per riferirsi alla sopraclasse
  - La classe Object è la sopraclasse di tutte le classi Java

V F  
V F  
V F  
V F  
V F  
V F

- 19 Completa le frasi seguenti utilizzando una tra le parole elencate alla fine della domanda.
- La parola chiave ..... viene utilizzata quando una variabile non fa riferimento ad alcun oggetto.
  - ..... è un riferimento implicito all'oggetto stesso.
  - ..... è un riferimento implicito alla sopraclasse dell'oggetto.
  - La ridefinizione di un metodo (ereditato) con lo scopo di modificarne il comportamento è indicata con il termine .....
  - La possibilità di utilizzare lo stesso nome per compiere operazioni diverse è indicata con il termine di .....
- this, new, public, private, null, overloading, super, void, overriding, information hiding*

- 20 Con quale strumento viene offerta la possibilità di creare nuove classi estendendo classi già esistenti?
- Incapsulamento
  - Ereditarietà
  - Polimorfismo
  - Astrazione

- 21 Completa le frasi seguenti utilizzando una tra le parole elencate alla fine della domanda.
  - a) La classe derivata da un'altra usando l'ereditarietà si chiama .....
  - b) La classe generatrice di una sottoclasse si chiama .....
  - c) Le relazioni sottoclasse-sopraclassi individuano una ..... di classi.
  - d) La gerarchia delle classi si può descrivere graficamente usando il .....  
**ereditarietà, sottoclasse, grafo di gerarchia, sopraclassi, base, diagramma delle classi, gerarchia, classe**
  
- 22 Quali sono i tipi di ereditarietà?
  - a) semplice e complessa
  - b) singola e doppia
  - c) semplice e doppia
  - d) singola e multipla
  
- 23 Quale tra queste definizioni corrisponde al termine *polimorfismo*?
  - a) La possibilità per i metodi di assumere forme diverse
  - b) La possibilità per un oggetto di avere diversi metodi
  - c) La possibilità di poter creare più oggetti diversi
  - d) La possibilità che una classe sia derivata da più sopraclassi

### Le librerie

- 24 Quale delle seguenti classi non è contenuta nel package *java.lang*?

- a) System
- b) Math
- c) String
- d) BufferedReader

- 25 Associa ad ogni metodo della classe *Math* della colonna di sinistra la corrispondente funzione calcolata scegliendola nella colonna di destra.

- |         |                         |
|---------|-------------------------|
| a) abs  | 1) elevamento a potenza |
| b) log  | 2) logaritmo            |
| c) pow  | 3) radice quadrata      |
| d) sqrt | 4) valore assoluto      |

- 26 Quali di queste affermazioni, riferite alle stringhe, sono vere (V) e quali false (F)?

- a) Fanno parte dei tipi di dato predefiniti di Java
- b) Sono gestite da un'apposita classe
- c) La classe *String* contiene vari metodi per manipolare le stringhe
- d) Ogni stringa ha una dimensione fissa

V F  
V F  
V F  
V F

## PROBLEMI

### Programmazione ad oggetti

- 1 Descrivere le caratteristiche dei seguenti oggetti: hard disk, computer, stampante.
- 2 Descrivere una radiosveglia indicando le sue caratteristiche e i suoi comportamenti.
- 3 Descrivere due o più oggetti *Televisore* usando il diagramma degli oggetti: indicare le caratteristiche e i valori associati.

- 4 Dall'elenco che segue, individuare gli oggetti e associare loro gli attributi e i metodi corretti: raggio; poligono; calcolaCirconferenza; numero lati; calcolaPerimetro; calcolaArea; cerchio.
- 5 Aggiungere altri attributi e altri metodi agli oggetti raggruppati nel problema precedente.
- 6 Data la seguente classe, creare tre istanze (oggetti) rappresentandole con il diagramma degli oggetti.

| <b>Pompa di carburante</b> |  |
|----------------------------|--|
| tipo carburante            |  |
| prezzo al litro            |  |
| capacità cisterna          |  |
| carburante rimasto         |  |
| modifica prezzo            |  |
| eroga carburante           |  |

- 7 Descrivere la classe *Televiore* usando il diagramma delle classi.
- 8 Usando il diagramma delle classi, descrivere un forno a microonde. Indicare tutti i possibili attributi e metodi completando il seguente diagramma.

| <b>Microonde</b> |  |
|------------------|--|
| marca            |  |
| ...              |  |
| impostaPotenza   |  |
| chiudiSportello  |  |
| ...              |  |
| ...              |  |

- 9 Usando il diagramma di classe *Microonde* descrivere graficamente il diagramma degli oggetti *forno1* e *forno2* attribuendo opportuni valori agli attributi.
- 10 Descrivere il diagramma di classe per modellare l'oggetto *mazzo di carte* pensando alle operazioni che potrebbe compiere un eventuale giocatore.
- 11 Utilizzando il diagramma delle classi, descrivere le caratteristiche di una figura geometrica piana e le trasformazioni elementari che possono essere ad essa applicate (affinità, omotetie, similitudini).
- 12 Utilizzando il diagramma delle classi, descrivere un dispositivo capace di trattare segnali elettrici (alimentatore, amplificatore, modulatore, campionatore, ecc.). Rappresentare graficamente, con il diagramma degli oggetti, due istanze della classe attribuendo opportuni valori agli attributi.

### Dichiarazione delle classi e uso degli oggetti

- 13 Descrivere la classe per il programma che calcola l'area e la circonferenza di un cerchio conoscendone il diametro.
- 14 Descrivere la classe per il programma che calcola l'area e il perimetro di un quadrato conoscendone il lato.

- 15 Descrivere la classe per il programma che calcola l'ipotenusa di un triangolo rettangolo, conoscendo il valore dei cateti.
  - 16 Descrivere la classe che simula una calcolatrice: deve essere in grado di memorizzare due numeri ed eseguire le operazioni elementari.
  - 17 Descrivere la classe Orologio che registra le ore e i minuti. Definire tre costruttori: il primo per inizializzare l'orologio a un'ora di default, il secondo per inizializzare solo le ore, il terzo per richiedere sia le ore che i minuti.
  - 18 Costruire una classe Atleta per rappresentare le informazioni di un giocatore.
  - 19 Definire la classe Email con gli attributi: destinatario, mittente, oggetto e testo. Implementare i metodi per inserire i precedenti attributi e per visualizzare il messaggio completo.
- Mascheramento dell'informazione e interazione tra oggetti**
- 20 Per ogni attributo della classe Orologio (creata nel Problema 17), scrivere un metodo per leggere il valore e un metodo per modificarlo.
  - 21 Scrivere un programma che memorizza il prezzo del carburante rilevato in cinque diversi distributori e successivamente calcola il prezzo massimo, minimo e medio.
  - 22 Scrivere un programma che memorizza il prezzo e la capacità (espressa in MB) di diversi supporti di memorizzazione (CD, DVD, chiavi USB). Successivamente si deve stabilire quale è il supporto che offre il minor costo per MB.
  - 23 Un compact disc può contenere dieci canzoni caratterizzate da un nome e una durata espressa in secondi. Scrivere una classe che descriva il CD e offra le seguenti operazioni:
    - modifica il titolo di una canzone
    - modifica la durata di una canzone
    - dato il nome di una canzone, restituisce la sua durata.
  - 24 Con riferimento alla classe creata nel problema precedente, scrivere un programma che, attraverso un menu, gestisce un CD consentendo di
    - inserire il nome e la durata delle canzoni
    - modificare il nome o la durata delle canzoni
    - data una canzone restituire la durata della stessa.
  - 25 Inserire in un vettore dieci libri, specificando per ognuno il titolo e il numero di pagine. Alla fine dell'inserimento, stampare il titolo di tutti i libri che hanno meno di 100 pagine.
  - 26 Scrivere un programma che consenta di manovrare un'automobile. Le due operazioni possibili sono: accelerare (A) e frenare (F). Il programma resta in attesa che l'utente inserisca un comando. Se inserisce il carattere A, l'automobile aumenta la velocità di 5 km/h. Se inserisce il carattere F l'automobile rallenta la velocità di 10 km/h. Dopo l'inserimento di ogni comando si deve mostrare la velocità attuale dell'automobile. Se si superano i 90 km/h deve essere segnalato un avvertimento: "Vai troppo forte. Rallenta". La velocità non può essere inferiore a zero. Inizialmente l'automobile sta viaggiando a 50 km/h.
  - 27 Ampliare il programma precedente aggiungendo la possibilità di spegnere (S) e accendere (I) la macchina. Se la macchina è spenta non è possibile né accelerare né frenare.

- ### Ereditarietà e polimorfismo
- 28 Descrivere la classe per il programma che calcola il volume di un parallelepipedo a base quadrata estendendo la classe di base Quadrato (creata nel Problema 14).
  - 29 Costruire una classe per rappresentare uno studente, estendendo la classe base Anagrafica (presentata nel Progetto 2) con un nuovo attributo per la matricola. Ridefinire il metodo per la stampa dei dati, visualizzando anche l'informazione sulla matricola.
  - 30 Costruire una classe per rappresentare un dipendente, estendendo la classe base Anagrafica, con gli attributi per lo stipendio e il livello (da 1 a 7). Prevedere un metodo per gestire l'incremento del livello: ad ogni scatto deve corrispondere un aumento del 10% dello stipendio.
  - 31 Con riferimento alla classe base creata nel problema precedente, derivare un nuovo dipendente specializzato. Per questo dipendente, il metodo per gestire l'incremento di livello viene sovrascritto considerando che, al raggiungimento del quinto livello, viene assegnato un premio di 1000 euro.
  - 32 Con riferimento alla classe base creata nel problema 16, derivare una calcolatrice per calcolare le seguenti funzioni: la somma dei quadrati dei due numeri, la differenza tra il quadrato del numero maggiore e il quadrato del minore.
  - 33 Con riferimento alla classe base creata nel problema 15, derivare una nuova classe Triangolo Colgorato con gli attributi per memorizzare il colore del bordo e il colore dello sfondo del triangolo. Definire un metodo con due parametri per impostare i colori del triangolo. Usando l'overloading, creare un altro metodo, con lo stesso nome del precedente, ma con un solo parametro per impostare il colore del bordo.
  - 34 Scrivere un programma per calcolare il costo di una telefonata da telefono fisso, conoscendo la sua durata in secondi e sapendo che il costo è di 15 centesimi al minuto. Definire un telefono cellulare, come estensione del telefono fisso, in cui il calcolo del costo è sovrascritto considerando una tariffa di 12 centesimi per lo scatto alla risposta e 0,35 centesimi al secondo.
- Le librerie**
- 35 Scrivere un programma che elenca le potenze di 2 a partire da  $2^0$  fino a  $2^{20}$ .
  - 36 Scrivere un metodo arrotonda che riceve come parametri un numero reale  $r$  e un numero intero  $i$ . Restituisce un numero reale che rappresenta l'arrotondamento del numero  $r$  alle prime  $i$  cifre decimali. Per esempio arrotonda(5.6794856,3) deve restituire il numero 5.679.
  - 37 Confrontare due stringhe che hanno lo stesso contenuto, per esempio "aaa" e "aaa", usando l'operatore ==. Rifare il confronto usando il metodo equals contenuto nella classe String.
  - 38 Dati dieci nomi di persona, contare e visualizzare solo quelli che iniziano con una vocale.
  - 39 Scrivere un programma che conta le lettere e i numeri contenuti in una stringa.

# ESERCIZI RISOLTI

Si è scelto di riprodurre il film per soli 5 minuti, per non incorrere in problemi con il produttore.  
In ogni caso, l'output ottenuto è il seguente:

Inizio proiezione di: Star Wars - Episodio III

Minuto: 1  
Minuto: 2  
Minuto: 3  
Minuto: 4  
Minuto: 5

Fine proiezione di: Star Wars - Episodio III

# ESERCIZI

## Vero o Falso?

- 1 I temini "classe" e "oggetto" sono sinonimi.  V  F
- 2 Una classe consente di creare oggetti.  V  F
- 3 Per ogni classe può esistere al massimo un oggetto.  V  F
- 4 Una classe non è un tipo di dato.  V  F
- 5 Un oggetto è un tipo di dato.  V  F
- 6 Una classe può non avere alcun metodo costruttore esplicito.  V  F
- 7 Il metodo costruttore serve per creare altre classi.  V  F
- 8 Il metodo costruttore deve avere lo stesso nome della classe a cui appartiene.  V  F
- 9 Il metodo costruttore deve essere solo uno.  V  F
- 10 In una stessa classe possono coesistere due metodi costruttori, purché abbiano numero e tipo di parametri differenti.  V  F
- 11 È importante sapere quali metodi costruttori possiede una classe, per poter creare un oggetto.  V  F
- 12 Gli attributi rappresentano le proprietà degli oggetti.  V  F
- 13 Utilizzare gli attributi in una classe ha senso solo se si la istanzia.  V  F
- 14 Un attributo static può essere letto da tutti gli oggetti di una classe.  V  F
- 15 Un attributo static può essere scritto solo da metodi static.  V  F
- 16 Un attributo final è condiviso da tutti gli oggetti di una classe.  V  F
- 17 Un attributo static final è una costante di classe.  V  F
- 18 I metodi possono essere considerati azioni che gli oggetti possono eseguire.  V  F
- 19 La firma di un metodo è ciò che lo distingue dagli altri.  V  F
- 20 L'overloading consente di implementare due metodi con la stessa firma in modo differente.  V  F
- 21 Un metodo static può essere invocato su un oggetto.  V  F
- 22 main è un metodo static.  V  F
- 23 Il metodo main deve essere utilizzato all'interno di tutte le classi e si può adottare al posto del metodo costruttore.  V  F
- 24 Una classe con il metodo main non può essere istanziata.  V  F
- 25 Il metodo main possiede un parametro di input.  V  F

## Corretto o Errato?

Prendendo in esame la classe Automobile, valuta le porzioni di codice mostrate di seguito e, per ciascuna, indica se è corretta o errata.

```
public class Automobile {
 private static final int numeroDiRuote = 4;
 private String marca = null;
 private String modello = null;
 private int litriDiBenzina = 0;
 private boolean accesa = false;

 public Automobile(String marca,
 String modello) {
 this.marca = marca;
 this.modello = modello;
 }

 public void avvia() {
 if(litriDiBenzina > 0) {
 accesa = true;
 }
 }

 public void spegni() {
 accesa = false;
 }

 public void faiBenzina(int litriDiBenzina) {
 this.litriDiBenzina += litriDiBenzina;
 }
}

1 Automobile a;
a.avvia(); C E

2 Automobile a;
a = new Automobile(); C E
```

# ESERCIZI

- 3 Automobile a;  
a = new Automobile("Ferrari", "F430");  
a.avvia(); C E
- 4 Automobile a;  
a = new Automobile("Ferrari", "F430");  
a.faiBenzina(100);  
a.accesa = true; C E
- 5 // aggiunta del codice seguente  
public static String getMarca() {  
 return marca;  
} C E
- 6 // aggiunta del codice seguente  
public static int numVeicoli = 0; C E
- 7 Automobile a;  
a = new Automobile();  
a.spegni(); C E
- 8 // aggiunta del codice seguente  
public Automobile() {  
 litriDiBenzina += 10;  
 this.avvia();  
} C E
- 9 int r = Automobile.numeroDiRuote; C E

## Scelta multipla

- 1 Un oggetto può esistere senza che esista una classe che lo definisce?  
A Sì  
B No  
C Solo se fa parte degli oggetti predefiniti del linguaggio
- 2 Una classe può essere priva di metodi costruttori espliciti?  
A Sì  
B Mai  
C Si, ma solo se ha tutti metodi statici
- 3 Una classe A può leggere un attributo privato della classe B?  
A Sì, mediante un metodo pubblico della classe B  
B No, mai  
C Sì
- 4 Un oggetto a può passare se stesso come parametro di un metodo invocato su un oggetto b?  
A Sì, mediante l'utilizzo del nome dato all'oggetto  
B Sì, ma solo se i due oggetti sono dello stesso tipo  
C Sì, utilizzando la parola chiave this
- 5 Come è possibile creare un oggetto?  
A Quando due variabili che contengono oggetti sono uguali?  
B Che cosa si intende per dominio applicativo?  
C Definisci il concetto di metodo costruttore.
- 6 In quali occasioni si è costretti a utilizzare la parola chiave this?  
A Quali sono i livelli di visibilità di membri public, protected e private?  
B In quali situazioni è opportuno utilizzare metodi statici?  
C In quali situazioni è impossibile utilizzare metodi statici?
- 7 Come è possibile passare argomenti a un programma Java?  
A In quale caso due metodi della stessa classe possono avere la stessa firma?  
B Come è possibile contare le istanze di una classe?  
C Definisci il concetto di information hiding.
- 8 Come è possibile fare comunicare due oggetti?  
A Come è possibile passare se stesso come parametro di un metodo invocato su un oggetto b?  
B Realizza i diagrammi UML delle classi che popolano i seguenti domini applicativi. Implementa poi le classi Java esplorando il tipo di visibilità per attributi e metodi (non è necessario implementare i metodi, ma è sufficiente indicarne la funzione).  
C Si vuole implementare l'anagrafica pazienti di un medico di famiglia, in modo tale che per ogni paziente siano memorizzati nome, cognome, uno o più indirizzi, uno o più numeri di telefono, data di nascita, codice della mutua e codice fiscale. Deve essere possibile distinguere la tipologia di paziente, in modo che ciascuno di essi sia classificato come un paziente ordinario, ovvero registrato proprio con questo medico, oppure un paziente saltuario, ovvero che si reca da questo medico solo quando è fuori sede per lavoro. Inoltre, si vuole fare in modo che il codice fiscale sia calcolato al momento della creazione dell'oggetto mediante un metodo apposito, che non deve poter essere invocato da altri oggetti.
- 9 Una grande azienda necessita di un software che gestisca la divisione dei reparti, l'appartenenza degli impiegati a tali reparti e la gestione dei progetti su cui gli impiegati lavorano. L'impiegato è identificato da nome, cognome e da un codice univoco all'interno dell'azienda. Il reparto è identificato dal nome. Ogni reparto può avere più di un impiegato, ma ogni impiegato appartiene a solo un reparto. Un progetto può essere seguito da diversi impiegati e ogni impiegato può seguire più progetti contemporaneamente.  
A Leggi attentamente le seguenti specifiche e implementa un'applicazione Java che raggiunga gli obiettivi indicati. Si implementi una classe Frazione che possa formalizzare la frazione nota in ambito matematico. Di seguito è elencato uno schema indicativo della classe, da completare opportunamente implementando i metodi, il cui significato è definito nei commenti che li precedono. Implementa anche una classe ProvaFrazione con il metodo main, che consenta di utilizzare Frazione e i suoi metodi a titolo di esempio.
- 10 public class Frazione {  
  
 private int num;  
 private int den;  
  
 Frazione() {  
 num=0;  
 den=1;  
 }  
  
 Frazione(int a) {  
 num=a;  
 den=1;  
 }  
  
 Frazione(int a, int b) {  
 num=a;  
 den=b;  
 riduci();  
 }  
  
 /\*  
 \* Imposta il numeratore  
 \*/  
 public void setNum(int n) {  
 //inserire il codice  
 }  
  
 /\*  
 \* Imposta il denominatore  
 \*/  
 public void setDen(int n) {  
 //inserire il codice  
 }  
  
 /\*  
 \* Restituisce il numeratore  
 \*/  
}

# ESERCIZI in Laboratorio

Realizza i diagrammi UML delle classi che popolano i seguenti domini applicativi. Implementa poi le classi Java esplorando il tipo di visibilità per attributi e metodi (non è necessario implementare i metodi, ma è sufficiente indicarne la funzione).

- 1 Si vuole implementare l'anagrafica pazienti di un medico di famiglia, in modo tale che per ogni paziente siano memorizzati nome, cognome, uno o più indirizzi, uno o più numeri di telefono, data di nascita, codice della mutua e codice fiscale. Deve essere possibile distinguere la tipologia di paziente, in modo che ciascuno di essi sia classificato come un paziente ordinario, ovvero registrato proprio con questo medico, oppure un paziente saltuario, ovvero che si reca da questo medico solo quando è fuori sede per lavoro. Inoltre, si vuole fare in modo che il codice fiscale sia calcolato al momento della creazione dell'oggetto mediante un metodo apposito, che non deve poter essere invocato da altri oggetti.

- 2 Una grande azienda necessita di un software che gestisca la divisione dei reparti, l'appartenenza degli impiegati a tali reparti e la gestione dei progetti su cui gli impiegati lavorano. L'impiegato è identificato da nome, cognome e da un codice univoco all'interno dell'azienda. Il reparto è identificato dal nome. Ogni reparto può avere più di un impiegato, ma ogni impiegato appartiene a solo un reparto. Un progetto può essere seguito da diversi impiegati e ogni impiegato può seguire più progetti contemporaneamente.

- 3 Leggi attentamente le seguenti specifiche e implementa un'applicazione Java che raggiunga gli obiettivi indicati. Si implementi una classe Frazione che possa formalizzare la frazione nota in ambito matematico. Di seguito è elencato uno schema indicativo della classe, da completare opportunamente implementando i metodi, il cui significato è definito nei commenti che li precedono. Implementa anche una classe ProvaFrazione con il metodo main, che consenta di utilizzare Frazione e i suoi metodi a titolo di esempio.

```
public class Frazione {

 private int num;
 private int den;

 Frazione() {
 num=0;
 den=1;
 }

 Frazione(int a) {
 num=a;
 den=1;
 }

 Frazione(int a, int b) {
 num=a;
 den=b;
 riduci();
 }

 /*
 * Imposta il numeratore
 */
 public void setNum(int n) {
 //inserire il codice
 }

 /*
 * Imposta il denominatore
 */
 public void setDen(int n) {
 //inserire il codice
 }

 /*
 * Restituisce il numeratore
 */
}
```

# ESERCIZI in Laboratorio

```

public double getNum() {
 //inserire il codice
}

/*
 * Restituisce il denominatore
 */
public double getDen() {
 //inserire il codice
}

/*
 * Moltiplica l'oggetto Frazione per un'altra frazione
 */
public void moltiplicaPer(Frazione f2) {
 //inserire il codice
}

/*
 * Divide l'oggetto Frazione per un'altra frazione
 */
public void dividiPer(Frazione f2) {
 //inserire il codice
}

/*
 * Somma l'oggetto Frazione a un'altra frazione
 */
public void sommaA(Frazione f2) {
 //inserire il codice
}

/*
 * Sottrae all'oggetto Frazione un'altra frazione
 */
public void sottrai(Frazione f2) {
 //inserire il codice
}

/*
 * Riduce la frazione
 */
private void riduci() {
 //inserire il codice
}

/*
 * Restituisce un oggetto Frazione, moltiplicazione tra quest'oggetto Frazione
 * e quello passato come parametro
 */
public Frazione per(Frazione f2) {
 //inserire il codice
}

/*
 * Restituisce il valore double di questa frazione
 */
public double valore() {
 //inserire il codice
}

```

# ESERCIZI in Laboratorio

```

/*
 * Restituisce true se quest'oggetto Frazione e' equivalente in valore a
 * quello passato come parametro, false altrimenti
 */
public boolean uguale(Frazione f2) {
 //inserire il codice
}

/*
 * Restituisce true se quest'oggetto
 * Frazione e' maggiore in valore a quello passato come parametro, false altrimenti
 */
public boolean maggiore(Frazione f2) {
 //inserire il codice
}

/*
 * Restituisce true se quest'oggetto
 * Frazione e' minore in valore a quello passato come parametro, false altrimenti
 */
public boolean minore(Frazione f2) {
 //inserire il codice
}

/*
 * Mostra a video il valore di questa frazione
 */
public void stampa() {
 //inserire il codice
}

/*
 * Restituisce un oggetto Frazione, moltiplicazione di 2 frazioni
 */
public static Frazione per(Frazione f1,Frazione f2) {
 //inserire il codice
}

/*
 * Restituisce un oggetto Frazione, divisione di 2 frazioni
 */
public static Frazione diviso(Frazione f1,Frazione f2) {
 //inserire il codice
}

/*
 * Restituisce un oggetto Frazione, differenza di 2 frazioni
 */
public static Frazione meno(Frazione f1,Frazione f2) {
 //inserire il codice
}

/*
 * Restituisce un oggetto Frazione, somma di 2 frazioni
 */
public static Frazione piu(Frazione f1,Frazione f2) {
 //inserire il codice
}

```

# ESERCIZI in Laboratorio

```
/*
 * Restituisce una stringa nel formato "(num/den)"
 */
public String toString() {
 //inserire il codice
}
```

- 4 Si vuole progettare una semplice applicazione che implementi la comunicazione tra un computer e un monitor. Il contenuto della comunicazione deve essere fornito al main dell'applicazione sotto forma di una stringa, e verrà trasmessa dall'oggetto computer all'oggetto monitor un byte alla volta. L'oggetto monitor dovrà essere in grado di visualizzare le informazioni trasmesse.
- 5 Implementa il classico gioco del tris, in modo che sia giocato dal computer contro se stesso. Progetta quindi una classe Computer che sia in grado di effettuare una scelta della casella da segnare con una "X" o con una "O", a seconda del simbolo a essa assegnato. La scelta della casella può essere effettuata utilizzando la classe java.util.Random per generare un numero casuale compreso tra 1 e 9. Il main dell'applicazione deve prendere in input i nomi da assegnare ai due oggetti computer che giocheranno l'uno contro l'altro.
- 6 JavaChannel, una nuova emittente televisiva, necessita di un sistema per far giungere il proprio segnale ai suoi abbonati. Implementa un'applicazione che consenta a un oggetto di tipo JavaChannel di comunicare il segnale agli oggetti di tipo Abbonato. Questi, ogni volta che ricevono il segnale, dovranno mostrarlo a video. Il segnale può essere qualsiasi tipo di informazione, per esempio una serie di stringhe statiche inviate una alla volta. Come argomenti, il main deve prendere in input il numero degli abbonati e i nomi di ciascuno di essi in modo che siano identificabili.

Unità di apprendimento

## B2

# Gli oggetti: concetti avanzati

### SOMMARIO

- B2.1 Ereditarietà**
  - B2.1.1 Un esempio di ereditarietà**
  - B2.1.2 Definizione di sottoclassi**
- B2.2 Gerarchia di classi**
- B2.3 Ereditarietà singola e multipla**
- B2.4 Estensione e ridefinizione**
  - B2.4.1 super**
  - B2.4.2 La classe Object**
  - B2.4.3 final**
- B2.5 Classi e metodi astratti**
- B2.6 Vantaggi dell'ereditarietà**
- B2.7 L'interfaccia verso il mondo esterno**
  - B2.7.1 Utilizzo del polimorfismo mediante l'interfaccia**

### PAROLE CHIAVE

- abstract
- Ereditarietà
- Ereditarietà multipla
- Ereditarietà singola
- Estensione e ridefinizione
- final
- Gerarchia di classi
- Polimorfismo
- Sottoclasse
- super
- Superclasse

### OBIETTIVI

- ✓ Comprendere i vantaggi offerti dal meccanismo dell'ereditarietà
- ✓ Sapere utilizzare l'ereditarietà per progettare le proprie classi
- ✓ Conoscere il concetto di astrazione
- ✓ Conoscere le interfacce
- ✓ Sapere quando utilizzare interfacce, classi astratte e metodi astratti nei propri progetti

## B2.1 Ereditarietà

L'ereditarietà è un'importante funzionalità della programmazione orientata agli oggetti che consente di definire nuove classi partendo da altre sviluppate in precedenza. La nuova classe è definita esprimendo solo le differenze che essa presenta rispetto alla classe di partenza.

Fino a questo momento abbiamo costruito le classi "partendo da zero", definendone cioè tutte le caratteristiche; l'ereditarietà consente di specificare "il punto di partenza", ovvero la classe base, e le differenze rispetto a questa.

Quando potremmo avere bisogno di questo costrutto di programmazione? Che cosa ci consente di fare?

Nella programmazione orientata agli oggetti capita di dover formalizzare oggetti del mondo reale che sono simili tra loro o, per meglio dire, sono addirittura della stessa "specie".

Per quanto riguarda la specie possiamo fare un esempio con gli animali; in particolare, consideriamo il regno animale e classifichiamolo in modo semplificato in uccelli, pesci e mammiferi.

In generale si potrebbe definire una classe Animale con alcune proprietà, quali

- colore degli occhi;
- peso;

## QUESITI

### 1 Il bytecode è ...

- A ... il codice sorgente di un programma Java.
- B ... il codice eseguibile generato dal compilatore Java.
- C ... un codice intermedio generato dal compilatore Java che deve essere interpretato dalla JVM.
- D Nessuna delle risposte precedenti.

### 2 Quali delle seguenti affermazioni sono vere a proposito della Java Virtual Machine (JVM)?

- A È indipendente dalla piattaforma hardware/software.
- B Ogni tipo di piattaforma hardware/software ha una specifica JVM.
- C La JVM è di fatto un interprete del bytecode.
- D La JVM è un componente hardware.

### 3 La portabilità del codice Java è data dal fatto che ...

- A ... il bytecode sviluppato su una qualsiasi piattaforma hardware/software può essere eseguito su una qualsiasi piattaforma hardware/software.
- B ... un programma sorgente Java può essere compilato con un qualsiasi compilatore, anche di un altro linguaggio di programmazione.
- C ... che la JVM sia sempre la stessa su qualsiasi piattaforma hardware/software.
- D Nessuna delle risposte precedenti.

### 4 Il motto «Compile once, run anywhere» significa che ...

- A ... il bytecode prodotto dalla compilazione Java può essere utilizzato su qualunque piattaforma identica alla piattaforma di compilazione indipendentemente dalla JVM.
- B ... il bytecode prodotto dalla compilazione Java può essere utilizzato su qualunque piattaforma hardware/software solo se fornita di JVM.
- C ... il bytecode prodotto dalla compilazione Java può essere utilizzato su qualunque piat-

taforma hardware solo se ha lo stesso sistema operativo della piattaforma di compilazione e se fornita di JVM.

D ... il bytecode prodotto dalla compilazione Java può essere utilizzato su qualunque piattaforma software solo se ha lo stesso hardware della piattaforma di compilazione e se fornita di JVM.

### 5 Una classe è ...

- A ... una collezione di oggetti: collegando gli oggetti tra di loro si implementa l'incapsulamento.
- B ... un modello da cui si creano oggetti simili.
- C ... l'implementazione di un oggetto nella sintassi Java.
- D Nessuna delle risposte precedenti.

### 6 Due oggetti diversi istanza della stessa classe ...

- A ... condividono solo il valore degli attributi.
- B ... condividono solo i metodi.
- C ... condividono metodi e struttura generale.
- D ... condividono codice dei metodi e tipo degli attributi.

### 7 Quali delle seguenti affermazioni a proposito di un oggetto sono vere?

- A Le proprietà di un oggetto descrivono il suo stato.
- B I metodi di un oggetto descrivono il suo stato.
- C I metodi si riferiscono alle funzionalità di un oggetto.
- D Un oggetto è un'astrazione di un oggetto reale.

### 8 Attraverso quale meccanismo gli oggetti interagiscono tra di loro?

- A Lo scambio di messaggi attraverso l'invocazione dei metodi.
- B Lo scambio di metodi attraverso l'inoltro di messaggi.
- C La condivisione del valore degli attributi.
- D Nessuna delle risposte precedenti.

### 9 Quale è la funzione dell'operatore Java new?

- A Creare una classe.
- B Creare un metodo.
- C Creare un oggetto.
- D Creare un package.

### 10 È possibile creare due riferimenti distinti a uno stesso oggetto?

- A Sì, sempre.
- B No, mai.
- C Sì, ma solo se i riferimenti sono entrambi dello stesso tipo dell'oggetto che riferiscono.
- D Sì, ma solo se i riferimenti non sono entrambi dello stesso tipo dell'oggetto che riferiscono.

### 11 Nel linguaggio Java lo spazio di memoria assegnato agli attributi di un oggetto istanziato l'operatore new viene liberato ...

- A ... automaticamente dal supporto a tempo di esecuzione del linguaggio.
- B ... solo al termine dell'esecuzione del programma.
- C ... automaticamente dal distruttore della classe.
- D Nessuna delle risposte precedenti.

### 12 Quali delle seguenti è la forma generale per applicare un metodo a un oggetto?

- A nomeOggetto.nomeMetodo(...)
- B nomeMetodo(...).nomeOggetto
- C NomeClasse:nomeMetodo(...)
- D nomeMetodo(nomeOggetto)

### 13 Collegare i seguenti tipi predefiniti del linguaggio Java con la relativa implementazione:

|        |                       |
|--------|-----------------------|
| int    | integer 8 bit         |
| short  | integer 16 bit        |
| long   | integer 32 bit        |
| float  | integer 64 bit        |
| double | floating-point 32 bit |
| char   | floating-point 64 bit |
| byte   | Unicode               |

### 14 Quali delle seguenti affermazioni a proposito del metodo main di una classe sono vere?

- A È il primo metodo da cui si avvia l'esecuzione del codice di una classe.
- B Il primo metodo che viene applicato ogni volta che si istanzia un oggetto.
- C È un metodo statico.
- D Non prevede parametri.

### 15 Quali delle seguenti affermazioni a proposito di un costruttore sono vere?

- A È un metodo che viene eseguito automaticamente all'atto della creazione di un oggetto.
- B Può prevedere come parametro un oggetto.
- C Può non essere pubblico.
- D Non prevede tipo di ritorno perché implicitamente restituisce un oggetto della classe.

### 16 Il valore degli attributi dichiarati nella sezione privata di una classe ...

- A ... può essere modificato mediante specifici metodi della classe stessa.
- B ... può essere modificato solo dal costruttore.
- C ... non può essere modificato.
- D ... può essere modificato direttamente con un'espressione del tipo oggetto.attributo = espressione;

### 17 Quali delle seguenti affermazioni circa un attributo con accessibilità di livello protected sono vere?

- A La sua accessibilità è equivalente a public.
- B La sua accessibilità diretta è possibile solo a livello di classe e classe derivata.
- C La sua accessibilità diretta è possibile solo a livello di classe e di package.
- D La sua accessibilità diretta è possibile solo a livello di classe, classe derivata e di package.

### 18 I membri statici di una classe sono ...

- A ... le costanti.
- B ... gli attributi e i metodi che appartengono non alle singole istanze, ma alla classe.

- C ... membri che possono essere riferiti con una notazione del tipo `NomeClasse.nomeMembro`.  
D Nessuna delle risposte precedenti.

19 Le classi *wrapper* dei tipi di dato primitivi sono ...

- A ... gli stessi tipi di dato primitivi.  
B ... classi che permettono di gestire valori associati ai tipi di dato primitivi come fossero oggetti.  
C ... classi che permettono a oggetti di qualunque classe di gestire tipi di dato primitivi.  
D ... classi che hanno come metodi gli operatori algebrici per definire espressioni con i tipi di dato primitivi.

20 Indicare quanto vale in Java l'output generato dalla seguente istruzione

```
System.out.println(3+4+" = "+3+4);
```

- A 7 = 7  
B 7 = 34  
C 34 = 7  
D 34 = 34

21 Una stringa di caratteri in Java è ...

- A ... un tipo di dato primitivo.  
B ... un *array* di caratteri.  
C ... un oggetto costante di classe `String`.  
D ... un oggetto variabile di classe `String`.

22 Javadoc è ...

- A ... una classe i cui oggetti permettono di generare documenti.  
B ... uno strumento Java per la documentazione automatica di programmi tramite *tag* inseriti nei commenti del codice da parte del programmatore.  
C ... un insieme di specifiche a cui attenersi per scrivere programmi aderendo alle convenzioni di stile del linguaggio Java.  
D Nessuna delle risposte precedenti.

23 Javabeans è ...

- A ... una classe i cui oggetti permettono di sviluppare documenti.

- B ... uno strumento Java per la documentazione automatica dei programmi.  
C ... una classe Java che osserva alcune convenzioni di codifica.  
D ... la classe da cui derivano tutte le classi del linguaggio Java.

## ESERCIZI

1 Si intende definire una classe `Persona` che abbia come attributi età, nome, sesso, professione. Realizzare il diagramma UML della classe e la relativa implementazione in linguaggio Java prevedendo oltre ai metodi *getter* e *setter* per ogni attributo:

- un costruttore che permetta di istanziare oggetti di tipo `Persona` definendo valori specifici per i vari attributi;
- il metodo `chiSei` che restituisca, quando invocato, una stringa del tipo «Sono una persona di nome: *nome*, sesso: *sesso*, età: *età*, professione: *professione*» dove *nome*, *sesso*, *età* e *professione* sono i valori assunti dagli attributi della classe;
- il metodo `main` che preveda la creazione di un oggetto di tipo `Persona` e l'invocazione del metodo `chiSei` per visualizzarne il risultato restituito.

2 Si devono gestire degli oggetti di tipo `Angolo` nella forma  $g^{\circ}m's''$  dove *g* rappresenta i gradi, *p* i primi e *s* i secondi (con  $0 \leq g < 360$ ,  $0 \leq p < 60$ ,  $0 \leq s < 60$ ). Dopo avere prodotto il diagramma UML della classe `Angolo` la si implementi in linguaggio Java rendendo disponibili i seguenti metodi (*n* rappresenta un valore intero, mentre *a* rappresenta un oggetto di tipo `Angolo`):

|                                   |                                                                                          |
|-----------------------------------|------------------------------------------------------------------------------------------|
| <code>visualizzaAngolo()</code>   | visualizza l'angolo nel formato $g^{\circ}m's''$ .                                       |
| <code>aggiungiGradi(n)</code>     | aggiunge all'angolo <i>n</i> gradi                                                       |
| <code>aggiungiPrimi(n)</code>     | aggiunge all'angolo <i>n</i> primi                                                       |
| <code>aggiungiSecondi(n)</code>   | aggiunge all'angolo <i>n</i> secondi                                                     |
| <code>angoloSecondi()</code>      | ritorna il valore dell'angolo espresso in secondi                                        |
| <code>secondiAngolo(n)</code>     | imposta il valore dell'angolo (gradi, primi e secondi) corrispondente a <i>n</i> secondi |
| <code>differenzaSecondi(a)</code> | restituisce la differenza espressa in secondi tra l'angolo e l'angolo <i>a</i>           |
| <code>sommaAngolo(a)</code>       | somma all'angolo l'angolo <i>a</i>                                                       |

Si definisca inoltre un costruttore che accetti in ingresso tre parametri interi per il valore dei gradi, dei primi e dei secondi e un costruttore di copia. Quando aggiornati i valori degli attributi devono essere normalizzati rispettando i limiti imposti (360 per i gradi, 60 per i primi e i secondi). Implementare infine il metodo `main` dove sono istanziate due o più oggetti di tipo `Angolo`, in modo da verificare l'invocazione di ogni singolo metodo in diverse condizioni.

3 Progettare mediante un diagramma UML e implementare in linguaggio Java una classe i cui oggetti rappresentano programmi per computer. Ogni oggetto deve avere almeno i seguenti attributi: *denominazione*, *produttore*, *versione*, *sistema operativo*, *anno*. La classe deve avere almeno i seguenti metodi:

- costruttore che ha come parametri *denominazione*, *produttore*, *versione*, *sistema operativo*, *anno*;
- costruttore di copia;
- `getDenominazione`, `getProduttore`, `getVersione`, `getSistema` e `getAnno` che restituiscono i valori degli attributi relativi;
- `setDenominazione`, `setProduttore`, `setVersione`, `setSistema` e `setAnno` che modificano i valori degli attributi relativi;
- `toString` che restituisce una stringa con tutti i dati dell'oggetto su cui è invocato;
- `compareAnno` che consente di confrontare l'anno di rilascio del programma con l'anno di rilascio di un altro programma.

Inserire nella classe un metodo `main` che consenta di verificare tutte le funzionalità.

4 Progettare mediante un diagramma UML e implementare in linguaggio Java una classe `CD` i cui oggetti rappresentano CD audio. Ogni oggetto `CD` deve avere almeno le seguenti caratteristiche: *titolo*, *autore*, *numero brani*, *durata*. La classe deve avere i seguenti metodi:

- costruttore che ha come parametri *titolo*, *autore*, *numero brani* e *durata*;
- `getTitolo`, `getAutore`, `getDurata` e `getBrani` che restituiscono i valori degli attributi relativi;
- `setTitolo`, `setAutore`, `setDurata` e `setBrani` che modificano i valori degli attributi relativi;

- `toString` che restituisce una stringa con tutti i dati dell'oggetto su cui è invocato;  
• `compareDurata` che consente di confrontare la durata complessiva del `CD` con la durata complessiva di un altro `CD`.

Inserire nella classe un metodo `main` che consenta di verificare tutte le funzionalità.

5 Una catena di autonoleggio deve gestire con un sistema informatico i propri veicoli; per ogni veicolo devono essere memorizzate le seguenti informazioni:

- targa;
- marca;
- modello;
- cilindrata;
- anno di acquisto;
- numero di posti.

Definire mediante un diagramma UML e implementare in linguaggio Java una classe per rappresentare gli oggetti di tipo `Veicolo` che rispetti le specifiche *Javabeans* prevedendo un metodo `main` per testarne le funzionalità.

6 Una organizzazione deve catalogare i computer utilizzati dai propri dipendenti; per ogni computer devono essere memorizzate le seguenti informazioni:

- codice;
- marca;
- modello;
- velocità del processore;
- dimensioni della memoria RAM;
- dimensioni del disco;
- dimensioni del monitor;
- anno di acquisto.

Il codice di ogni computer è un numero progressivo generato automaticamente e non ulteriormente modificabile. Definire mediante un diagramma UML e implementare in linguaggio Java una classe per rappresentare gli oggetti di tipo `Computer` che rispetti le specifiche *Javabeans* prevedendo un metodo `main` per testarne le funzionalità.

7 Progettare mediante un diagramma UML e implementare, mediante una classe `Java` che rispetti le specifiche *Javabeans*, una classe per la rappresentazione di un punto sulla superficie

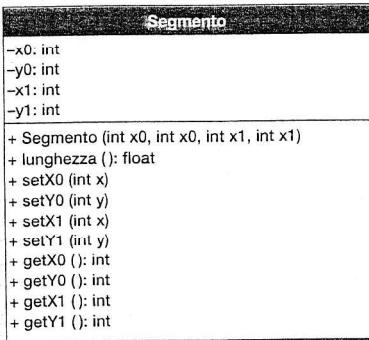


FIGURA 13

della Terra individuato dalle proprie coordinate geografiche (latitudine e longitudine espresse in gradi, primi e secondi).

- 8** Progettare mediante un diagramma UML e implementare in linguaggio Java una classe per la gestione di rettangoli da visualizzare su un display LCD monocromatico di 800 × 600 pixel (i rettangoli hanno i lati paralleli ai bordi del display e di essi sono visualizzati solo i lati) ipotizzando di disporre della seguente classe Segmento di FIGURA 13.

La classe *Rettangolo* deve esporre – oltre ai metodi *setter* e *getter* – metodi per:

- calcolare l'area in pixel del rettangolo;
- determinare se il rettangolo interseca o meno un secondo rettangolo;
- spostare il rettangolo lungo le direzioni parallele ai lati del display;
- ruotare il rettangolo di 90° rispetto al proprio centro.

La classe deve inoltre disporre di un metodo *main* che consenta un test adeguato delle proprie funzionalità.

- 9** Modificare le classi *Segmento* e *Rettangolo* di cui all'esercizio precedente in modo da adattarle per l'uso con un display LCD a colori; ipotizzare attributi e metodi della classe *Colore*.

- 10** Un vettore nel piano cartesiano è un segmento orientato definito dalle coordinate dei due estremi («origine» e «vertice») e dal loro ordinamento: due vettori con estremi coincidenti diversamente orientati sono sovrapposti, ma diversi! Implementare in linguaggio Java la classe di FIGURA 14, definita mediante un diagramma UML e relativa a un vettore nel piano cartesiano.

Inserire nella classe un metodo *main* che consenta di verificarne tutte le funzionalità.

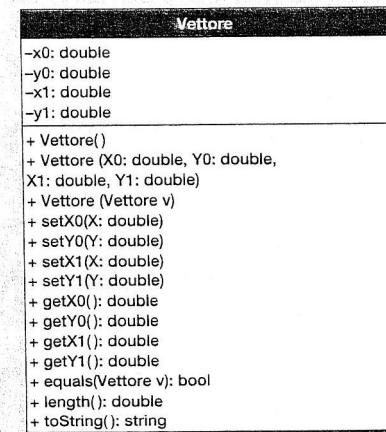


FIGURA 14

## ORIGINAL DOCUMENT

### 5.6 The standardised String objects

#### 5.6.1 Declaring and assigning String variables

In Java, a string of characters is manipulated with an object of type *String*. The *String* type does *not* belong to the primitive types (*boolean*, *int*, *double*, etc.), but is rather a non-primitive object type. Therefore, a variable of type *String* is a reference to that object in the global memory. Thus for the same reasons as for the case of integers, the following *badSwap* function will not swap the string variables since once the function is executed the values of *p* and *q* (that is, the respective references) are kept unchanged: Java is *passby-value/reference* only. For non-primitive types such as arrays, objects and strings, the stored value is a reference.

```

static void badSwap(String p, String q)
{
 String tmp;
 tmp=p;
 p=q;
 q=tmp;
}

```

To declare and initialise a *String*, we proceed as follows

```

String title="First Hands-on Programming!";
String anotherTitle=title;
...
title="Extreme programming sounds better!"; // String can be modified
To print a String, use the regular System.out.println() function:
System.out.println(title);

```

Java proposes many standard functions for efficiently manipulating strings. Let us next review only the very fundamental functions that operate on *String* objects.

#### 5.6.2 Length of a string: *length()*

The length of a *String* object is defined as its number of characters. The length of a *String* is reported by calling the *length()* method on the *String* object, as shown below:

```

String s="Supercalifragilisticexpialidocious";
System.out.println(s.length());

```

We get 34, which is the longest invented word in a song from *Mary Poppins* written by the Sherman Brothers in 1964. Observe that the syntax is different from arrays. For an array, we get its *length* (that is, its number of elements) by the following syntax: *array.length*. But for a *String*, we use the method *length()* (with no argument, which explains the *()* parentheses).

#### 5.6.3 Equality test for strings: *equals(String str)*

To test whether two *String* *s1* and *s2* are identical or not, do not use the regular *==* equality test for primitive types. Indeed, testing whether two strings are identical or not by using the comparison *==* will check whether the references of these strings are identical or not. Instead, to test whether the contents of the two strings are identical or not, use the method *equals(str)* with a *String* argument *str*. Consider for example, the following sequence of instructions:

```

String s1="java";
String s2="JAVA";
System.out.println(s1.equals(s2));

```

**passby-value**  
letteralmente «passare accanto a». Per una variabile *String* è rappresentato non tanto da suo valore effettivo ma solo dal riferimento per giungere a esso.

**Mary Poppins**  
personaggio dell'omonimo film del 1964 diretto da Robert Stevenson e basato sui romanzi scritti da Pamela Lyndon Travers.

**lexicographic**  
quello lessicografico è un criterio di ordinamento di stringhe equivalente a quello utilizzato nei dizionari e che può essere esteso a un qualsiasi insieme di simboli.

**span**  
distanza, intervallo.

**Costruttore di copia.** Se gli attributi di una classe non sono di un tipo di dato primitivo o immutabile (come per esempio gli oggetti di classe *String*), ma riferimenti a oggetti, è necessario che il codice del costruttore di copia li cloni senza limitarsi a copiarne i riferimenti; in caso contrario il costruttore crea un nuovo oggetto che condivide con l'oggetto originale i riferimenti agli attributi, e ogni variazione dell'uno si rifletterebbe in una automatica e spesso inaspettata variazione dell'altro. La stessa attenzione deve essere posta ogni volta che un metodo di una classe restituisce un riferimento a un attributo, o ha come parametro un oggetto da riferire da parte di un attributo: è sempre necessario effettuare una clonazione dell'oggetto.

**Eccezioni.** Sono eventi che si presentano in fase di esecuzione (*run-time*) di un programma al verificarsi di situazioni anomale (divisione per zero, uso di un indice fuori dal *range* di un *array*, accesso a un riferimento nullo, ...). Un'eccezione può essere intercettata e gestita, oppure il programma termina la sua esecuzione. L'intercettazione e la gestione delle eccezioni si basa sul costrutto *try/catch*. La parola chiave *try* permette di definire un blocco di istruzioni di cui controllare l'esecuzione per intercettare eventuali eccezioni che potrebbero verificarsi; ogni singola clausola *catch* definisce un blocco di istruzioni che devono recuperare la situazione anomala connessa alla specifica eccezione intercattata. In Java le eccezioni sono oggetti istanza di classi che derivano dalla classe predefinita *Exception*, che a sua volta deriva da *Throwable*; il linguaggio definisce alcune classi di eccezioni predefinite: *ArithmaticException*, *ArrayIndexOutOfBoundsException*, *NullPointerException*, *IOException*, .... Una classe eccezione può essere minimale come la seguente

```
public class Eccezione extends Exception {};
```

oppure il programmatore può definire attributi, costruttori e metodi per le necessità di gestione degli oggetti che costituiscono le eccezioni concrete sollevate, in particolare per fornire informazioni specifiche sul tipo di errore che ha causato l'eccezione stessa.

**Generazione di eccezioni.** Per segnalare il proprio insuccesso, un metodo di una classe Java può generare un'eccezione. Un metodo che

può generare un'eccezione deve specificare questa eventualità nella propria firma, utilizzando la parola chiave *throws* seguita dal tipo di eccezione, o da un elenco di tipi di eccezioni, che possono essere sollevate dal metodo. Nel codice del metodo la parola chiave *throw* consente di interrompere l'esecuzione e di sollevare l'eccezione specificata come un nuovo oggetto. La semantica dell'istruzione *throw* ha alcuni aspetti in comune con quella dell'istruzione *return*: in particolare l'esecuzione di *throw* implica la terminazione immediata del metodo e il passaggio del controllo al codice chiamante del metodo stesso. Un metodo può sollevare più tipi di eccezione, che possono essere singolarmente gestite specificando clausole *catch* multiple.

**Handle or declare.** Questa regola («gestisci o dichiara») stabilisce che, a fronte di una possibile eccezione di tipo controllato (*checked*), il codice di un metodo deve intercettarla e gestirla, oppure dichiarare a sua volta di sollevarla; un'eccezione non deve mai passare inosservata: se un metodo non la gestisce, trasferisce l'obbligo di gestirla al metodo che lo ha invocato. Alcune eccezioni predefinite di classe *RuntimeException* sono di tipo non controllato (*unchecked*) e possono essere non gestite e non dichiarate: in questo caso la loro eventuale generazione causa l'interruzione dell'esecuzione del programma.

**Input/output predefinito.** Gli oggetti predefiniti *System.out* di classe *PrintStream* e *System.in* di classe *InputStream* sono normalmente associati rispettivamente allo standard output (finestra di tipo testuale sullo schermo) e standard input (tastiera). I metodi che rendono disponibili consentono di eseguire operazioni di input/output tradizionali (per esempio utilizzando i metodi *print* o *println* della classe *PrintStream*): la complessità della classe *InputStream* può essere gestita incapsulandolo in un oggetto di tipo *BufferedReader* che rende disponibile il metodo *readLine* per la lettura di una riga di testo dallo standard input.

**Input/output sequenziale da file di testo.** Le classi *BufferedReader* e *BufferedWriter*, che incapsulano a loro volta oggetti di tipo *FileReader* e *FileWriter*, rendono disponibili semplici metodi per la lettura/scrittura di singole righe di un file di testo.

**Serializzazione e persistenza degli oggetti.** La serializzazione di un oggetto è un processo che permette di salvarlo in un supporto di memorizzazione sequenziale (come un file), o di trasmetterlo su un canale di comunicazione sequenziale (come una connessione di rete). La serializzazione può essere effettuata in forma binaria, oppure può impiegare codifiche testuali come il formato XML (*eXtensible Markup Language*). Lo scopo della serializzazione è quello di salvare e/o trasmettere l'intero stato dell'oggetto, in modo che esso possa essere successivamente ricreato nello stesso identico stato dal processo inverso, spesso denominato *deserializzazione*. Una classe i cui oggetti devono essere serializzati deve implementare l'interfaccia *Serializable* del package *java.io*:

```
public class ClasseSerializzabile
 implements java.io.Serializable {
 ...
}
```

Se tale classe ha come attributi dei tipi di dato non primitivi, anche le classi di cui sono istanza devono a loro volta implementare l'interfaccia *Serializable*. Un oggetto serializzabile può essere reso persistente su file invocando il metodo *writeObject* della classe  *ObjectOutputStream*, che incapsula un oggetto di tipo  *FileOutputStream*. L'oggetto può essere ripristinato dal file invocando il metodo *readObject* della classe  *ObjectInputStream*, che incapsula un oggetto di tipo  *FileInputStream*.

## QUESITI

**1 In Java gli array sono oggetti?**

- A No, mai.
- B Si in ogni caso.
- C Si, ma solo se non contengono tipi di dato primitivi.
- D Nessuna delle risposte precedenti.

**2 Un elemento di un array di oggetti è vuoto se ...**

- A ... non contiene nulla.
- B ... contiene il valore 0.
- C ... contiene il valore predefinito *null*.
- D ... contiene il riferimento a un oggetto creato con il costruttore di default.

**3 Con l'istruzione *int[] unVettore;* ...**

- A ... si istanzia un vettore di valori numerici interi.
- B ... si dichiara un riferimento a un vettore di valori numerici interi.
- C ... si istanzia un vettore di *??????*.
- D ... si dichiara un vettore di riferimenti a oggetti di tipo *int*.

**4 Ponendo uguale a *null* un elemento di un array di riferimenti a oggetti ...**

- A ... l'oggetto a cui faceva riferimento viene distrutto.
- B ... l'oggetto a cui faceva riferimento viene distrutto solo se non è riferito da altre variabili.
- C ... continua a esistere in ogni caso.
- D Nessuna delle risposte precedenti.

**5 L'attributo *length* di un array contiene ...**

- A ... il numero di elementi dell'*array*.
- B ... il numero di elementi non vuoti dell'*array*.
- C ... il numero di elementi vuoti dell'*array*.
- D ... il numero di byte occupato dall'*array* nell'area di memoria *heap*.

**6 Le eccezioni sono ...**

- A ... eventi anomali rilevati dal compilatore in fase di generazione del *byte-code*.
- B ... eventi anomali rilevati dall'ambiente di esecuzione a *run-time*.
- C ... avvisi normalmente generati dall'ambiente di esecuzione.
- D Nessuna delle risposte precedenti.

**7** La gestione delle eccezioni avviene in Java tramite il costrutto ...

- A ... if/else.
- B ... try/catch.
- C ... switch/case.
- D ... do/while.

**8** Indicare quali delle seguenti affermazioni sono vere rispetto alla generazione delle eccezioni.

- A Se un metodo genera un'eccezione deve anche gestirla.
- B Un metodo che genera un'eccezione deve contenere la clausola throws nella sua firma.
- C Un'eccezione può essere generata da un metodo utilizzando l'istruzione throw.
- D Un'eccezione può essere generata da un metodo utilizzando l'istruzione try.

**9** Indicare quali delle seguenti affermazioni sono false rispetto al sollevamento delle eccezioni.

- A L'istruzione throw sostituisce il costrutto try/catch.
- B Nel codice di un metodo che genera eccezioni non è necessaria l'istruzione return.
- C Per generare specifici tipi di eccezioni definite dall'utente è necessario definire una classe ad hoc.
- D Un metodo non può generare più di un tipo di eccezioni.

**10** Indicare quali delle seguenti affermazioni circa le eccezioni sono vere.

- A Le eccezioni non necessariamente sono oggetti.
- B Tutte le classi definite per rappresentare le eccezioni derivano gerarchicamente dalla classe Java Throwable.
- C Tutte le classi definite per rappresentare le eccezioni derivano gerarchicamente dalla classe Java Exception.
- D Tutte le classi definite per rappresentare le eccezioni derivano gerarchicamente dalla classe Java RuntimeException.

**11** La regola «Handle or declare» si riferisce al fatto che ...

- A ... tutti gli oggetti per essere gestiti (handle) devono essere preventivamente dichiarati (declare).

B ... a fronte di una possibile eccezione un metodo deve gestirla (handle), oppure dichiarare a sua volta di generarla (declare).

C ... a fronte di una possibile eccezione un metodo deve gestirla (handle) e dichiarare di generarla (declare).

D Nessuna delle risposte precedenti.

**12** Dati i metodi insert1 e insert2 aventi come scopo l'inserimento di un valore numerico intero in una specifica posizione del vettore vector, descritto dal frammento di codice riportato a fianco, indicare quali delle seguenti affermazioni sono vere.

- A Il primo metodo genera un'eccezione, mentre il secondo no.
- B Il primo metodo non genera un'eccezione, mentre il secondo sì.
- C Il primo metodo intercetta un'eccezione, mentre il secondo no.
- D Il primo metodo non intercetta un'eccezione, mentre il secondo sì.

**13** Serializzare un oggetto significa ...

- A ... memorizzarlo di seguito ad altri oggetti dello stesso tipo.
- B ... memorizzarlo di seguito ad altri oggetti non necessariamente dello stesso tipo.
- C ... salvare un oggetto su un supporto di memorizzazione sequenziale per renderlo persistente, o trasmetterlo su un canale di comunicazione sequenziale per renderlo trasferibile.
- D Nessuna delle risposte precedenti.

**14** Quali delle seguenti affermazioni sono vere rispetto al processo di serializzazione/deserializzazione di un oggetto?

- A Permette di rendere un oggetto persistente.
- B La deserializzazione permette di ripristinare un oggetto precedentemente serializzato.
- C Una classe i cui oggetti devono essere serializzati deve implementare l'interfaccia Java Serializable.
- D Il formato della serializzazione può essere sia binario sia testuale.

```
...
private int[] vector;
...
public void insert1(int value, int position) throws ArrayIndexOutOfBoundsException {
 vector[position] = value;
 return;
}
...
public boolean insert2(int value, int position) {

 try {
 vector[position] = value;
 return true;
 }
 catch (ArrayIndexOutOfBoundsException exception) {
 return false;
 }
}
```

## ESERCIZI

**1** Facendo riferimento alla classe Programma progettata e implementata nell'esercizio 3 del capitolo precedente, progettare mediante un diagramma UML e implementare in linguaggio Java una classe i cui oggetti rappresentano dei contenitori, ciascuno dei quali può contenere fino a un massimo di  $N$  oggetti di tipo Programma. Dopo avere stabilito le strutture dati necessarie, si definiscono i seguenti metodi:

- costruttore avente come parametro il numero  $N$ ;
- getProgramma: ha come parametro la posizione del programma nel contenitore e restituisce un oggetto Programma corrispondente a quello contenuto nella posizione specificata;
- setProgramma: inserisce un programma in una specifica posizione del contenitore e ha come parametro un oggetto Programma;
- killProgramma: elimina il programma presente nella posizione specificata come parametro;
- getN: restituisce il numero di programmi presenti nel contenitore;
- cercaProgrammaPerDenominazione: restituisce la posizione nel contenitore del programma con la denominazione corrispondente se presente, -1 altrimenti;

- *toString*: restituisce una stringa contenente l'elenco delle denominazioni di tutti i programmi contenuti nel contenitore;
- *confrontaContenitore*: consente di confrontare i programmi di un contenitore con quelli di un diverso contenitore restituendo il numero di programmi in comune.

**2** Facendo riferimento alla classe CD progettata e implementata nell'esercizio 4 del capitolo precedente, progettare mediante un diagramma UML e implementare in linguaggio Java una classe PortaCD i cui oggetti rappresentano dei contenitori, ciascuno dei quali può contenere fino a un massimo di  $N$  oggetti di tipo CD. Dopo avere stabilito le strutture dati necessarie, si definiscono i seguenti metodi:

- costruttore avente come parametro il numero  $N$ ;
- *getCD*: ha come parametro la posizione del CD nel portaCD e restituisce un oggetto CD corrispondente a quello contenuto nella posizione specificata;
- *setCD*: inserisce un CD in una specifica posizione del portaCD e ha come parametro un oggetto CD;
- *killCD*: elimina il CD presente nella posizione specificata del portaCD come parametro;

- getN*: restituisce il numero di CD presenti nel portaCD;
- cercaCDperTitolo*: restituisce la posizione nel portaCD del CD con il titolo corrispondente se presente, -1 altrimenti;
- toString*: restituisce una stringa contenente l'elenco dei titoli di tutti i CD contenuti nel portaCD;
- controllaCollezione*: consente di confrontare i CD di un portaCD con quelli di una diversa collezione restituendo il numero di CD in comune.

**3** Modificare la classe *Mensola* presentata nel corso del capitolo per fare in modo che, ferma restando la capienza massima, i libri siano sempre disposti in modo compatto dalla parte sinistra della mensola senza spazi vuoti tra un libro e l'altro:

- inserire un libro in una determinata posizione comporta spostare tutti i libri a partire da tale posizione di un posto a destra;
- inserire un libro in una posizione a destra dell'ultimo libro presente comporta inserire comunque il nuovo libro nella posizione immediatamente a destra dell'ultimo libro;
- eliminare un libro in una determinata posizione significa spostare di una posizione verso sinistra tutti i libri alla sua destra.

**4** Un albergo ha un sistema di gestione delle chiavi delle camere automatizzato: i clienti prendono e restituiscono le chiavi da un sistema portachiavi, senza l'intervento del portiere. Ogni chiave è identificata dal numero della camera (non necessariamente progressivo) ed è associata al nominativo del cliente a cui è stata assegnata la camera. Il sistema portachiavi può contenere le chiavi di tutte le camere dell'albergo e ogni posizione può essere occupata con una qualsiasi chiave. Il cliente che esce dall'albergo lascia la propria chiave nella prima posizione libera, il cliente che entra nell'albergo richiede la chiave fornendo il numero della camera o il proprio nominativo. Si richiede di rappresentare la soluzione mediante un diagramma UML delle classi e di implementarlo in linguaggio Java prevedendo le necessarie eccezioni.

**5** Scrivere un metodo statico che – a partire dai valori dei coefficienti *a*, *b* e *c* di un'equazione di secondo grado – calcoli la prima soluzione gene-

rando una specifica eccezione nel caso di equazione impossibile.

**6** La seguente classe Java implementa un semplice *stack* di interi:

```
public class Stack {
 private int[] mem;
 private int p, n;

 public Stack(int N) {
 mem = new int[N];
 p = 0;
 n = 0;
 }

 public void push(int x) {
 mem[p] = x;
 p++;
 n++;
 }

 public int pop() {
 int x;

 x = mem[p-1];
 p--;
 n--;
 return x;
 }

 public int size() {
 return n;
 }
}
```

Riscrivere i metodi *pop* e *push* della classe *Stack* in modo che sollevino specifiche eccezioni rispettivamente per la situazione di *stack* vuoto (*empty*) e pieno (*full*).

**7** Una catena di autonoleggio deve gestire con un sistema informatico i propri veicoli; per ogni veicolo devono essere memorizzate le seguenti informazioni: codice, targa, marca e modello, numero di posti (si veda in proposito l'esercizio 5 del capitolo precedente). Si intende progettare una possibile soluzione per la gestione informatica di quasi 1000 veicoli avente le seguenti funzionalità:

- aggiunta di un nuovo veicolo (il codice deve essere un numero sequenziale incrementato

automaticamente ogni volta che si aggiunge un veicolo);

- eliminazione di un veicolo dato il codice o la targa;
- ricerca delle informazioni di un veicolo dato il codice o la targa;
- ricerca di tutti i veicoli aventi un dato numero di posti;
- salvataggio e ripristino su/da file dell'intero insieme di veicoli;
- effettuare l'inventario di quante macchine per ogni marca dispone l'autonoleggio, nella forma marca, numero veicoli.

a) Definire mediante un diagramma UML le classi che consentono di rappresentare adeguatamente la soluzione del problema.

b) Implementare in linguaggio Java la classi progettate prevedendo e sollevando specifiche eccezioni.

c) Scrivere un metodo *main* che consenta la gestione (aggiunta, eliminazione, ricerca per targa o per codice, elenco dato il numero di posti) dell'intero insieme di veicoli, visualizzando messaggi di errore in caso di sollevamento di eccezioni.

**8** Un porto turistico affitta i propri posti-barca (circa un centinaio) alle imbarcazioni che ne fanno richiesta. Per legge è tenuto a registrare per ogni barca ospitata le seguenti informazioni: nome, nazionalità, lunghezza, stazza, tipologia (vela o motore); ma non vi è obbligo di mantenere le informazioni relative alle imbarcazioni dopo che hanno lasciato il porto. I posti-barca sono numerati: i posti da 1 a 20 non possono ospitare barche più lunghe di 10 m e le barche a vela devono essere piazzate in via prioritaria nei posti successivi al 50. Il costo dell'affitto per le barche a vela è di 10 € per metro di lunghezza al giorno, mentre per le barche a motore è di 20 € per tonnellata di stazza al giorno. È richiesta la progettazione di una possibile soluzione per la gestione informatica dei posti-barca che implementi le seguenti funzionalità:

- assegnazione di un posto a una barca in arrivo;
- liberazione di un posto occupato con calcolo dell'importo dell'affitto (in input viene fornito il numero dei giorni di sosta);
- ricerca delle informazioni relative alla barca che occupa un dato posto;

- salvataggio su file dello stato del porto in un certo istante in modo da renderlo persistente;
- produrre una struttura dati (array) dei nomi delle barche di una certa nazionalità specificata dall'utente.

a) Definire mediante un diagramma UML le classi che consentono di rappresentare adeguatamente la soluzione del problema.

b) Implementare in linguaggio Java le classi progettate prevedendo e sollevando specifiche eccezioni.

c) Dotare la classe principale di un metodo *main* che permetta all'utente di esercitare le funzionalità elencate visualizzando messaggi di errore in caso di sollevamento di eccezioni.

**9** Una biblioteca scolastica deve gestire mediante un'applicazione software un elenco di circa 1000 libri: per ogni libro è necessario memorizzare l'autore, il titolo, l'anno di pubblicazione e l'editore. L'applicazione deve consentire le seguenti operazioni:

- aggiunta di un nuovo libro alla biblioteca;
- ricerca di un libro a partire dal titolo;
- ricerca di tutti i libri di uno specifico autore;
- determinazione del numero di libri presenti.

Si richiede di rappresentare la soluzione mediante un diagramma UML delle classi e di implementarla in linguaggio Java prevedendo la generazione delle opportune eccezioni.

**10** Una grande organizzazione deve catalogare i computer utilizzati dai propri dipendenti; per ogni computer devono essere memorizzate le seguenti informazioni: codice, marca, modello, velocità del processore, dimensioni della memoria RAM, dimensioni del disco, dimensioni del monitor e anno di acquisto. Il codice di ogni computer è un numero progressivo generato automaticamente e non ulteriormente modificabile. Dopo avere definito mediante un diagramma UML e implementato in linguaggio Java una classe per rappresentare gli oggetti di tipo *Computer* (si veda in proposito l'esercizio 6 del capitolo precedente), progettare in linguaggio UML e implementare in linguaggio Java una classe che consenta la gestione (aggiunta, eliminazione, ricerca per codice, ricerca di tutti i computer con caratteristiche migliori di valori di velocità e dimensione

di memoria/disco specificate, salvataggio e ripristino su/da file dell'intero catalogo), prevedendo e generando specifiche eccezioni.

## LABORATORIO

- 1** Si vuole simulare un cronometro che effettua la misura del tempo in secondi. Realizzare un diagramma UML e la relativa implementazione in linguaggio Java di una classe *Cronometro* assumendo che le operazioni possibili siano le seguenti:

```
init() inizializza un oggetto cronometro azzerando il suo accumulatore di secondi
start() avvia il cronometro che inizia a contare i secondi
stop() ferma il cronometro interrompendo il conteggio dei secondi
show() prevede un parametro che può assumere i valori «s», «m» o «h» in funzione del quale visualizza il tempo conteggiato in secondi dal cronometro rispettivamente in: secondi, minuti:secondi, ore:minuti:secondi
```

Per realizzare la classe *Cronometro* si utilizzi il metodo statico *System.currentTimeMillis*, che restituisce un valore di tipo *long* che rappresenta il numero di millisecondi trascorsi dalle ore 00:00 del 1/1/1970.

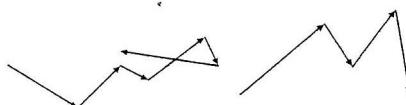
Dotare inoltre la classe di un costruttore che azzerà il cronometro e di un metodo *main* che istanzi due oggetti di tipo *Cronometro c1 e c2* e che visualizzi i risultati forniti dalla seguente sequenza di operazioni:

- start di c1
- attesa 5 secondi
- show in secondi del valore di c1
- start di c2
- attesa 61 secondi
- stop di c1
- show in minuti:secondi di c1
- attesa 4 secondi
- start di c1
- attesa 2 secondi
- stop di c2
- show di c2 in secondi
- stop di c1
- show di c1 in ore:minuti:secondi

Per realizzare il metodo *main* si implementi il seguente metodo statico avente lo scopo di sospendere l'esecuzione del codice di *n* secondi:

```
public static void attendiNsecondi
 (long n) {
 try {
 Thread.currentThread().sleep
 (n * 1000);
 }
 catch (InterruptedException exception) {
 }
}
```

- 2** Facendo riferimento alla classe *Vettore* implementata nell'esercizio 10 del capitolo precedente, aggiungere un metodo booleano *interseca* avente come parametro un altro oggetto *Vettore* e che restituisca *true* se i due vettori si intersecano, *false* altrimenti. Per implementare il metodo *interseca* si prenda in considerazione il codice C/C++ riportato a fianco.  
Nei sistemi CAD (*Computer Aided Design*) e GIS (*Geographic Information Systems*) una *polyline* è una sequenza ordinata di *N* vettori *v1, v2, v3, ..., vN* nel piano cartesiano concatenati in modo che l'origine di ciascuno (escluso il primo) coincida con il vertice del precedente, come illustrato negli esempi seguenti che riproducono rispettivamente una *polyline* intrecciata e una semplice:



Progettare – mediante un diagramma di classe UML – e implementare in linguaggio Java una classe *Polyline* che consenta di definire in sequenza tutti i vettori che costituiscono la *polyline* stessa e inoltre di eseguire le seguenti operazioni:

- calcolare la lunghezza totale della *polyline*;
- determinare se la *polyline* è semplice o intrecciata;
- salvare e ripristinare tutti i vettori che costituiscono la *polyline* in/da un file di tipo testuale.

- 3** Si intende realizzare un programma Java che consenta di gestire la navigazione in barca. Il programma deve permettere all'operatore di inserire manualmente le successive posizioni rilevate mediante un dispositivo GPS, che restituisce latitudine e longitudine in gradi, primi e secondi, e di calcolare in ogni momento la distanza percorsa

```
struct POINT
{
 float x ;
 float y;
};

struct SEGMENT
{
 POINT p1;
 POINT p2;
};

int ccw (POINT p0, POINT p1, POINT p2)
{
 int dx1, dx2, dy1, dy2;

 dx1 = p1.x - p0.x;
 dy1 = p1.y - p0.y;
 dx2 = p2.x - p0.x;
 dy2 = p2.y - p0.y;

 if (dx1*dy2 > dy1*dx2)
 return +1;
 if (dx1*dy2 < dy1*dx2)
 return -1;
 if ((dx1*dx2 < 0) || (dy1*dy2 < 0))
 return -1;
 if ((dx1*dx1+dy1*dy1) < (dx2*dx2+dy2*dy2))
 return +1;
 return 0;
}
```

```
bool intersect(SEGMENT s1, SEGMENT s2)
{
 return ((ccw(s1.p1,s1.p2,s2.p1)*ccw(s1.p1,s1.p2,s2.p2)) <=0) &&
 ((ccw(s2.p1,s2.p2,s1.p1)*ccw(s2.p1,s2.p2,s1.p2)) <=0);
}
```

[R. Sedgewick, *Algoritmi in C++*, Addison-Wesley, 1993]

dall'ultimo punto rilevato e dal punto di partenza. Il programma deve permettere inoltre di inserire una destinazione espressa mediante i valori della latitudine e della longitudine e di calcolare la rotta (cioè l'angolo rispetto al Nord) e la lunghezza del percorso rettilineo che congiunge l'ultimo punto rilevato con il punto di destinazione. Sono richiesti:

- un diagramma UML delle classi;
- un programma in linguaggio Java con interfaccia utente testuale.

Per adattare le coordinate geografiche (latitudine/longitude) in formato decimale in coordinate cartesiane Nord/Est espresse in metri, modificare il codice C/C++ riportato nella pagina seguente.

# Strutture dati

```

#define PI 3.141592653589793
#define WGS84_E2 0.006694379990197
#define WGS84_E4 WGS84_E2*WGS84_E2
#define WGS84_E6 WGS84_E4*WGS84_E2
#define WGS84_SEMI_MAJOR_AXIS 6378137.0
#define WGS84_SEMI_MINOR_AXIS 6356752.314245
#define UTM_LONGITUDE_OF_ORIGIN 3.0/180.0*PI
#define UTM_LATITUDE_OF_ORIGIN 0.0
#define UTM_FALSE_EASTING 500000.0
#define UTM_FALSE_NORTHING_N 0.0
#define UTM_FALSE_NORTHING_S 10000000.0
#define UTM_SCALE_FACTOR 0.9996

double m_calc(double latitude)
{
 return (1.0 - WGS84_E2/4.0 -
 3.0*WGS84_E4/64.0 -
 5.0*WGS84_E6/256.0) * latitude -
 (3.0*WGS84_E2/8.0 +
 3.0*WGS84_E4/32.0 +
 45.0*WGS84_E6/1024.0) * sin(2.0*latitude) +
 (15.0*WGS84_E4/256.0 +
 45.0*WGS84_E6/1024.0) * sin(4.0*latitude) -
 (35.0*WGS84_E6/3072.0) * sin(6.0*latitude);
}

// INPUT: position in latitude/longitude (WGS84)
// OUTPUT: position in UTM easting/northings (meters)
void GPS2UTM(double latitude, double longitude, double* easting, double* northing)
{
 int int_zone;
 double M, M_origin, A, A2, e2_prim, C, T, v;

 int_zone = (int)(longitude/6.0);
 if (longitude < 0)
 int_zone--;
 longitude -= (double)(int_zone)*6.0;
 longitude *= PI/180.0;
 latitude *= PI/180.0;
 M = WGS84_SEMI_MAJOR_AXIS*m_calc(latitude);
 M_origin = WGS84_SEMI_MAJOR_AXIS * m_calc(UTM_LATITUDE_OF_ORIGIN);
 A = (longitude - UTM_LONGITUDE_OF_ORIGIN) * cos(latitude);
 A2 = A*A;
 e2_prim = WGS84_E2/(1.0 - WGS84_E2);
 C = e2_prim*pow(cos(latitude), 2.0);
 T = tan(latitude);
 T *= T;
 v = WGS84_SEMI_MAJOR_AXIS /
 sqrt(1.0 - WGS84_E2 * pow(sin(latitude), 2.0));
 northing = UTM_SCALE_FACTOR(M - M_origin + v*tan(latitude) *
 (A2/2.0 + (5.0 - T + 9.0*C + 4.0*C*C) *
 A2*A2/24.0 + (61.0 - 58.0*T + T*T + 600.0*C -
 330.0*e2_prim)*A2*A2*A2/720.0));
 if (latitude < 0)
 *northing += UTM_FALSE_NORTHING_S;
 *easting = UTM_FALSE_EASTING + UTM_SCALE_FACTOR*v *
 (A + (1.0 - T + C)*A2*A/6.0 +
 (5.0 - 18.0*T + T*T + 72.0*C - 58.0*e2_prim) *
 A2*A2*A/120.0);
}

```

Si deve realizzare un programma per la gestione dell'elenco degli invitati a una festa dove il numero di invitati è variabile: a seconda delle necessità dovremo aggiungere nuove persone a cui inizialmente non avevamo pensato, oppure eliminarne altre che hanno declinato l'invito. In casi di questo tipo l'utilizzo di un *array* non offre la necessaria flessibilità: infatti fissare a priori un determinato numero di elementi può comportare un eccesso di elementi inutilizzati, oppure trovarsi senza posizioni disponibili se il numero degli elementi cresce oltre il previsto.

È necessario realizzare una struttura dati di dimensione variabile in cui sia possibile inserire o eliminare elementi in una posizione qualsiasi. Schematicamente un struttura dati di questo tipo può essere rappresentata come in FIGURA 1, dove:

- *head* («testa») è il riferimento di ingresso alla struttura;
- ogni elemento, denominato «nodo», della struttura è un oggetto ripartito logicamente in due componenti: una informativa (*info*), che in questo caso contiene il riferimento ai dati di un *Invitato* (*Invitato 1*, *Invitato 2*, ... *Invitato n*), e una che realizza il collegamento all'*Invitato* successivo (*link*);
- l'ultimo nodo ha la componente *link* con valore *null*;
- per scorrere la lista è necessario seguire la catena dei riferimenti (l'accesso è strettamente sequenziale);
- è possibile inserire o eliminare nodi alla lista inserendoli/eliminandoli in una posizione qualsiasi (all'inizio, al termine, in una posizione intermedia) modificando in modo opportuno le componenti *link* dei nodi.

Per inserire, ad esempio, un nuovo invitato come secondo elemento della lista (magari per collocarlo vicino a un altro invitato che conosce) si opera come segue (FIGURA 2):

1. Si crea un nuovo nodo la cui componente *info* riferisce il *Nuovo Invitato*.

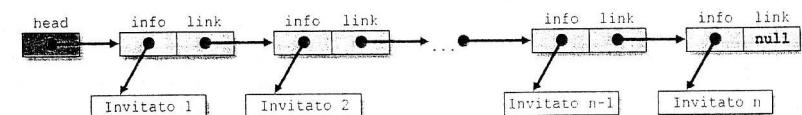


FIGURA 1



# ESERCIZI

## Vero o Falso?

Nel caso in cui S sia sottoclasse della classe base B, rispondi "Vero" o "Falso" alle seguenti domande.

- 1 Un oggetto s della classe S può essere utilizzato in ogni situazione in cui è richiesto un oggetto b di B.  V  F
- 2 Un oggetto b della classe B può essere utilizzato in ogni situazione in cui è richiesto un oggetto s di S.  V  F
- 3 Tutti i metodi di S sono ereditati NELLA classe B.  V  F
- 4 Tutti i metodi di B sono ereditati NELL'A classe S.  V  F
- 5 In S si possono sovrascrivere tutti i metodi di B.  V  F
- 6 La sovrascrittura di un metodo è detta *overloading*.  V  F
- 7 Un metodo definito final in B non può essere ridefinito in S.  V  F
- 8 Nella classe B è possibile accedere a tutti i metodi della classe S.  V  F
- 9 super è una parola riservata che permette di fare riferimento agli attributi privati della classe stessa.  V  F
- 10 Nella classe S è possibile accedere direttamente agli attributi di B anche se questi sono definiti privati.  V  F
- 11 Nella classe B mediante la parola riservata this è possibile accedere agli attributi e ai metodi di S, purché questi non siano definiti privati.  V  F
- 12 In S si possono sovrascrivere tutti i metodi di B purché questi non siano definiti final.  V  F
- 13 Se la classe B è definita final non è possibile la relazione di ereditarietà.  V  F
- 14 Se la classe B è definita final è possibile la relazione di ereditarietà solo se anche S è definita final.  V  F
- 15 I costruttori di B sono accessibili da S mediante la parola riservata super.  V  F
- 16 S deve ridefinire obbligatoriamente tutti i costruttori di B.  V  F
- 17 In S è possibile ridefinire un metodo di B purché se ne cambi la firma.  V  F
- 18 Ogni classe deriva implicitamente dalla classe Object.  V  F
- 19 Ridefinendo il metodo *toString* si elimina la dipendenza gerarchica di una classe dalla classe Object.  V  F

Digita qui il testo

- 20 La classe Object è definita final.  V  F
- 21 Non è possibile istanziare un oggetto b della classe B se la classe B è astratta.  V  F
- 22 Se almeno un metodo di B è astratto allora la classe B è astratta.  V  F
- 23 Se la classe B è definita astratta e S non ridefinisce tutti i metodi astratti di B allora S è a sua volta astratta.  V  F
- 24 Se S ridefinisce tutti i metodi astratti di B allora è possibile istanziare un oggetto s della classe S.  V  F
- 25 S non può ridefinire i metodi di B se questi sono astratti.  V  F
- 26 In una classe astratta non si possono definire metodi concreti.  V  F
- 27 L'interfaccia è l'insieme delle dichiarazioni dei metodi.  V  F
- 28 Un'interfaccia deve contenere almeno un metodo astratto.  V  F
- 29 L'interfaccia definisce anche gli attributi ma senza inizializzarli.  V  F
- 30 Una classe C che implementa un'interfaccia I è anche di tipo I.  V  F

## Corretto o Errato

Nel caso in cui Romanzo sia sottoclasse della classe base Libro, controlla le seguenti porzioni di codice e definisci se sono corrette o errate.

- 1 Romanzo r;  
Libro l;  
r=new Romanzo();  
l=r;  C  E
- 2 Romanzo r;  
Libro l;  
r=new Libro();  
l=new Libro();  C  E
- 3 Romanzo r;  
Libro l;  
r=new Romanzo();  
l=new Libro();  C  E
- 4 Romanzo r;  
Libro l;  
r=new Romanzo();  
l=new Romanzo();  C  E
- 5 Romanzo r1,r2;  
Libro l;  
r1=new Romanzo();  
l=r1;  
r2=l;  C  E
- 6 Romanzo r1,r2;  
Libro l;  
r1=new Romanzo();  
l=r1;  
r2= (Romanzo) l;  C  E

# ESERCIZI

## Scelta multipla

- 1 In che modo una sottoclasse può differenziarsi da una superclasse?
  - a Solo per estensione
  - b Solo per ridefinizione di alcuni metodi
  - c Per estensione e/o per ridefinizione
- 2 In Java è possibile l'ereditarietà multipla?
  - a In nessun modo
  - b Sì, con la sintassi class S extends B1, B2
  - c Sì, ma in modo indiretto utilizzando le interfacce
- 3 La modifica di un metodo in una superclasse si ripercuote nel comportamento di una sottoclasse?
  - a Solo se questo metodo non è stato ridefinito
  - b No, mai
  - c Solo se questo metodo non è stato ridefinito e nella sottoclasse non è presente un riferimento esplicito al metodo della superclasse mediante la parola chiave super
  - d Sì, sempre
- 4 Per implementare un'interfaccia si deve
  - a semplicemente utilizzare la parola chiave implements seguita dal nome dell'interfaccia
  - b utilizzare implements seguita dal nome dell'interfaccia e implementarne almeno un metodo
  - c utilizzare extends seguita dal nome dell'interfaccia e implementarne tutti i metodi
  - d utilizzare implements seguita dal nome dell'interfaccia e implementarne tutti i metodi

## Quesiti ed esercizi

- 1 In che cosa consiste il processo di astrazione per estensione?
- 2 In che cosa consiste il processo di astrazione per ridefinizione?
- 3 Che cos'è il polimorfismo?
- 4 Qual è la differenza tra le parole riservate this e super?
- 5 Definisci il concetto di metodo astratto.
- 6 Definisci il concetto di classe astratta.
- 7 Che cosa si intende con il termine overriding?
- 8 Che cosa si intende per interfaccia?
- 9 Tra le seguenti coppie di classi, stabilisci quali sono le superclassi e le sottoclassi:
  - Moto, MotoDaCorsa
  - Elettrodomestico, Televisore
  - CanzoneMp3, File
  - Atleta, Calciatore
  - Portiere, Calciatore
  - Studente, Universitario
- 10 Fai un esempio di superclasse e sottoclasse utilizzando il diagramma UML delle classi.
- 11 Fai un esempio di classe derivata per estensione.
- 12 Fai un esempio di classe derivata per ridefinizione.
- 13 Rappresenta una gerarchia di classi in cui è presente ereditarietà multipla.
- 14 Rappresenta una gerarchia di classi in cui è presente ereditarietà singola.
- 15 Rappresenta l'insieme di classi riportato di seguito definendo una gerarchia; si tratta di ereditarietà singola o multipla?
 

Studente, PersonaleDellaScuola, StudenteBiennio, Docente, StudenteTriennio, StudenteDellaMiaClasse
- 16 Dell'insieme di tipi che segue, individua quello che dovrebbe rappresentare un'interfaccia e definiscine alcuni metodi che possono essere implementati dalle altre classi: Autocarro, Motocicletta, Automobile, Veicolo, Locomotiva.

# ESERCIZI in Laboratorio

Scrivi le classi Java necessarie alla soluzione dei seguenti problemi.

- 1 La classe Rettangolo ha due attributi privati di tipo double: base e altezza; oltre al costruttore senza parametri che istanzia un rettangolo di base 0 e altezza 0, è presente un costruttore Rettangolo(double, double) i cui due parametri rappresentano il valore della base e dell'altezza del rettangolo. Sono presenti i metodi double Area() e double Perimetro() oltre al metodo toString().  
Definisci la classe derivata Quadrato con due costruttori, uno senza parametri e uno in cui viene passata la lunghezza del lato del quadrato. Ridefinisci i metodi che devono essere modificati.
- 2 La classe Dvd ha due attributi privati: titoloFilm e Genere; implementa la classe inserendo costruttori, metodi set e get oltre ai metodi toString e visualizza. Definisci la classe derivata DvdInNegozio con l'attributo Prezzo e tutti i metodi necessari.
- 3 Un negozio vende apparecchiature informatiche, in particolare computer, stampanti e scanner. Ogni apparecchiatura è identificata da un codice e da una descrizione; per i computer deve essere rappresentato il processore; per le stampanti laser deve essere rappresentata la velocità in pagine al minuto; per le stampanti a getto d'inchiostro deve essere definito il numero di colori e la risoluzione di stampa, mentre per gli scanner è importante definire la risoluzione in punti per pollice. Realizza i diagrammi UML delle classi esplicitando la struttura gerarchica.
- 4 Un supermercato fornisce ai suoi clienti una tessera che consente loro di accumulare punti in base alle spese effettuate. Il cliente può scegliere all'interno di un catalogo vari premi, ciascuno dei quali ha un "costo" definito in termini di punti. Realizza la classe Tessera che memorizza i dati del cliente (codice, cognomeNome e il numero di punti accumulati). Al momento del rilascio della tessera viene assegnato un punteggio di partenza di 10. Il metodo acquisto riceve come parametro l'importo della spesa e incrementa il punteggio di un punto ogni 10 € di spesa. Il metodo ritiraPremio riceve come parametro il numero di punti associati al premio e scala questi dal punteggio accumulato. Non esiste altro modo per modificare il punteggio, ma è possibile conoscere il numero di punti accumulati mediante il metodo getPunteggio. Al posto della normale tessera può essere rilasciata al cliente una tessera speciale denominata TesseraOro del tutto analoga alla precedente eccetto per il fatto che non riceve punti omaggio al momento del rilascio e non incrementa il punteggio per spese inferiori ai 100 €; per tutti gli importi superiori ai 100 € il cliente riceve 2 punti ogni 10 € di spesa. Implementa le due classi utilizzando il meccanismo dell'ereditarietà.
- 5 Facendo riferimento all'esercizio precedente, implementa la classe TesseraPlatino che ha un comportamento analogo a TesseraOro ma che non assegna punti per spese inferiori ai 200 €, mentre per importi superiori assegna 3 punti ogni 10 € di spesa.
- 6 Facendo riferimento all'esempio di fine unità già risolto, immagina che il negozio venga, oltre ai cd musicali e ai film in dvd, anche film in vhs e musicassette. Riprogetta la struttura delle classi ottimizzando la gerarchia con interfacce e/o classi astratte e definisci anche i metodi play(), stop(), pause(), rew() e ffw() dove necessario. Si noti che un'eventuale classe astratta generica, come Articolo nella soluzione proposta, non deve essere istanziabile, in quanto solo un oggetto specifico come cd, dvd, vhs o mc deve poter essere istanziabile.