Esercizi di riepilogo e approfondimento

** **R14.1.** Una fattura contiene una raccolta di articoli acquistati. Dovrebbe essere implementata mediante una lista o un insieme? Fornite spiegazioni.

** **R14.2.** Considerate un programma che gestisca un'agenda di appuntamenti. Dovrebbe inserirli in una lista, in una pila, in una coda o in una coda prioritaria? Fornite spiegazioni.

*** **R14.3.** Una possibile implementazione di un'agenda prevede l'utilizzo di una mappa che metta in corrispondenza oggetti di tipo "data", usati come chiavi, e oggetti di tipo "evento". Questo, però, funziona soltanto se è previsto un unico evento per ogni possibile data. Quale altro tipo di raccolta si può utilizzare per consentire la presenza di più eventi associati a una stessa data?

** **R14.4.** Analizzate, nella documentazione dell'interfaccia `Collection`, le descrizioni dei metodi `addAll`, `removeAll`, `retainAll` e `containsAll`. Descrivete come li si possa utilizzare per implementare le comuni operazioni tra insiemi (unione, intersezione, differenza e sottoinsieme).

* **R14.5.** Spiegate che cosa viene visualizzato dal codice seguente. Tracciate uno schema della lista concatenata dopo ciascun passo.

```
LinkedList<String> staff = new LinkedList<>();
staff.addFirst("Harry");
staff.addFirst("Diana");
staff.addFirst("Tom");
System.out.println(staff.removeFirst());
System.out.println(staff.removeFirst());
System.out.println(staff.removeFirst());
```

* **R14.6.** Spiegate che cosa viene visualizzato dal codice seguente. Tracciate uno schema della lista concatenata dopo ciascun passo.

```
LinkedList<String> staff = new LinkedList<>();
staff.addFirst("Harry");
staff.addFirst("Diana");
staff.addFirst("Tom");
System.out.println(staff.removeLast());
System.out.println(staff.removeFirst());
System.out.println(staff.removeLast());
```

* **R14.7.** Spiegate che cosa viene visualizzato dal codice seguente. Tracciate uno schema della lista concatenata dopo ciascun passo.

```
LinkedList<String> staff = new LinkedList<>();
staff.addFirst("Harry");
staff.addLast("Diana");
staff.addFirst("Tom");
System.out.println(staff.removeLast());
System.out.println(staff.removeFirst());
System.out.println(staff.removeLast());
```

* **R14.8.** Spiegate che cosa viene visualizzato dal codice seguente. Tracciate uno schema della lista concatenata e la posizione dell'iteratore dopo ciascun passo.

```
LinkedList<String> staff = new LinkedList<>();
ListIterator<String> iterator = staff.listIterator();
iterator.add("Tom");
iterator.add("Diana");
iterator.add("Harry");
```

```
iterator = staff.listIterator();
if (iterator.next().equals("Tom")) { iterator.remove(); }
while (iterator.hasNext()) { System.out.println(iterator.next()); }
```

★ **R14.9.** Spiegate che cosa viene visualizzato dal codice seguente. Tracciate uno schema della lista concatenata e la posizione dell'iteratore dopo ciascun passo.

```
LinkedList<String> staff = new LinkedList<>();
ListIterator<String> iterator = staff.listIterator();
iterator.add("Tom");
iterator.add("Diana");
iterator.add("Harry");
iterator = staff.listIterator();
iterator.next();
iterator.next();
iterator.add("Romeo");
iterator.next();
iterator.add("Juliet");
iterator = staff.listIterator();
iterator.next();
iterator.remove();
while (iterator.hasNext()) { System.out.println(iterator.next()); }
```

★★ **R14.10.** Data una lista concatenata di stringhe, come vi si eliminano tutti gli elementi che hanno lunghezza non superiore a tre?

★★ **R14.11 (per Java 8).** Risolvete nuovamente l'esercizio precedente usando il metodo `removeIf`, dopo averne letto la descrizione nella documentazione API dell'interfaccia `Collection`. Usate un'espressione lambda (descritta nella sezione Note per Java 8 10.4).

★★ **R14.12.** Quali vantaggi e svantaggi hanno le liste concatenate rispetto agli array?

★★ **R14.13.** Supponete di dover organizzare un elenco di numeri di telefono per un reparto di una società. Al momento vi sono circa 6000 dipendenti, sapete che il centralino può gestire al massimo 10000 numeri di telefono e prevedete che l'elenco venga consultato diverse centinaia di volte al giorno. Per memorizzare le informazioni usereste un vettore o una lista concatenata?

★★ **R14.14.** Immaginate di dover gestire un elenco di appuntamenti. Usereste una lista concatenata o un vettore di oggetti di tipo `Appointment`?

★ **R14.15.** Supponete di scrivere un programma che simula un mazzo di carte. Le carte vengono pescate dalla cima del mazzo e distribuite ai giocatori. Quando le carte tornano nel mazzo, vengono poste al di sotto del mazzo stesso. Memorizzereste le carte in una pila o in una coda?

★ **R14.16.** Ipotizzate che le stringhe `"A"` ... `"Z"` vengano inserite in una pila. Successivamente, vengono estratte dalla pila e inserite in una seconda pila. Infine, vengono estratte dalla seconda pila e visualizzate. In quale ordine?

★ **R14.17.** Qual è la differenza fra un insieme e una mappa?

★★ **R14.18.** L'unione di due insiemi A e B è l'insieme di tutti gli elementi che sono contenuti in A, in B o in entrambi. L'intersezione è, invece, l'insieme di tutti gli elementi che sono contenuti sia in A sia in B. Come si possono calcolare l'unione e l'intersezione di due insiemi, usando i metodi `add` e `contains`, oltre a un iteratore?

★★ **R14.19.** Come si possono calcolare l'unione e l'intersezione di due insiemi usando i metodi forniti dall'interfaccia `java.util.Set`, ma senza usare un iteratore? Consultate la documentazione API dell'interfaccia nella libreria standard.

* **R14.20.** Una mappa può avere due chiavi associate allo stesso valore? E due valori associati alla stessa chiave?

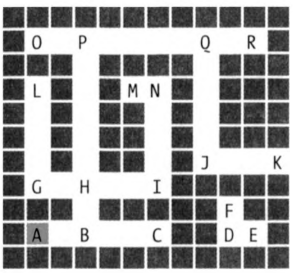
** **R14.21.** Una mappa può essere realizzata mediante un insieme di coppie (*chiave*, *valore*). Date una spiegazione.

* **R14.22 (per Java 8).** Come si possono visualizzare tutte le coppie chiave/valore presenti in una mappa usando il metodo `keySet`? E usando il metodo `entrySet`? E il metodo `forEach` fornendo un'espressione lambda (descritta nella sezione Note per Java 8 10.4)?

*** **R14.23.** Verificate che il codice di hash della stringa `"Juliet"` sia quello riportato nella Tabella 6.

*** **R14.24.** Verificate che le stringhe `"VII"` e `"Ugh"` abbiano lo stesso codice di hash.

* **R14.25.** Considerate l'algoritmo di uscita da un labirinto visto nel Paragrafo 14.6.4, immaginando di partire dalla posizione A e di inserire nella pila le direzioni nell'ordine ovest, sud, est e nord. In quale ordine vengono visitate le posizioni indicate dalle lettere nel labirinto qui rappresentato?



* **R14.26.** Ripetete l'esercizio precedente, usando una coda invece di una pila.

Esercizi di programmazione

** **E14.1.** Scrivete un metodo

```
public static void downsize(LinkedList<String> employeeNames, int n)
```

che elimini da una lista concatenata un impiegato ogni `n`.

** **E14.2.** Scrivete un metodo

```
public static void reverse(LinkedList<String> strings)
```

che inverta i dati presenti in una lista concatenata.

** **E14.3.** Realizzate il *crivello di Eratostene*, un metodo per calcolare i numeri primi noto agli antichi greci. Scegliete un numero n: questo metodo calcolerà tutti i numeri primi fino a n. Come prima cosa inserite in un insieme tutti i numeri da 2 a n. Poi, cancellate tutti i multipli di 2 (eccetto 2); vale a dire 4, 6, 8, 10, 12, ... Dopodiché, cancellate tutti i multipli di 3 (eccetto 3), cioè, 6, 9, 12, 15, ... Arrivate fino a $n^{1/2}$, quindi visualizzate l'insieme.

** **E14.4.** Scrivete un programma che usi una mappa in cui sia le chiavi sia i valori sono stringhe: rispettivamente, i nomi degli studenti e i loro voti in un esame. Chiedete all'utente del programma se vuole inserire o rimuovere studenti, modificarne il voto o stampare tutti i voti. La visualizzazione dovrebbe essere ordinata per nome e avere un aspetto simile a questo:

```
Carl: B+
Joe: C
Sarah: A
```

★★★ **E14.5.** Scrivete un programma che legga un file di codice sorgente Java e generi un elenco di tutti gli identificatori presenti, visualizzando, accanto a ciascuno di essi, i numeri delle righe in cui compare. Per semplicità considereremo che qualsiasi stringa costituita soltanto da lettere, cifre numeriche e caratteri di sottolineatura sia un identificatore. Dichiarate la variabile `Scanner in` per leggere il file e invocate il metodo `in.useDelimiter("[^A-Za-z0-9_]+")`, in modo che ogni invocazione di `next` restituisca un identificatore.

★★ **E14.6 (per Java 8).** Leggete da un file tutte le parole presenti e aggiungetele a una mappa le cui chiavi siano le lettere iniziali delle parole e i cui valori siano insiemi contenenti le parole che iniziano con quella stessa lettera. Quindi, visualizzate gli insiemi di parole in ordine alfabetico.

Risolvete l'esercizio in due modi, uno che usi il metodo `merge` (descritto nella sezione Note per Java 8 14.1) e uno che aggiorni la mappa come nella sezione Esempi completi 14.1.

★★ **E14.7 (per Java 8).** Leggete da un file tutte le parole presenti e aggiungetele a una mappa le cui chiavi siano le lunghezze delle parole e i cui valori siano stringhe composte da parole separate da virgole, con parole aventi tutte la stessa lunghezza. Quindi, visualizzate tali stringhe in ordine crescente di lunghezza delle loro singole parole componenti.

Risolvete l'esercizio in due modi, uno che usi il metodo `merge` (descritto nella sezione Note per Java 8 14.1) e uno che aggiorni la mappa come nella sezione Esempi completi 14.1.

★★ **E14.8.** Usate una pila per invertire le parole di una frase. Continuate a leggere parole, aggiungendole alla pila, fin quando non trovate una parola che termina con un punto. A questo punto estraete tutte le parole dalla pila e visualizzatele, poi ripetete la procedura fino all'esaurimento dei dati in ingresso. Ad esempio, questa frase

```
Mary had a little lamb. Its fleece was white as snow
```

deve essere trasformata nella seguente

```
Lamb little a had mary. Snow as white was fleece its.
```

Fate attenzione alle lettere maiuscole e al posizionamento del punto che termina la frase.

★ **E14.9.** Dovete scomporre nu numero intero nelle sue singole cifre, trasformando, ad esempio, il numero 1729 nella sequenza di cifre 1, 7, 2 e 9. L'ultima cifra del numero `n` si ottiene facilmente calcolando `n % 10`, ma procedendo in questo modo si ottengono le cifre in ordine inverso. Risolvete il problema usando una pila. Il programma deve chiedere all'utente di fornire un numero intero, per poi visualizzarne le singole cifre separate da spazi.

★★ **E14.10.** In occasione di manifestazioni particolari, il proprietario di una casa noleggia posti auto nel suo vialetto di casa, che può essere rappresentato da una pila, con il consueto comportamento "last-in, first-out". Quando il proprietario di un'automobile se ne va e la sua automobile non è l'ultima, tutte quelle che la bloccano devono essere spostate temporaneamente sulla strada, per poi rientrare nel vialetto. Scrivete un programma che simuli questo comportamento, usando una pila per il vialetto e una per la strada, con numeri interi a rappresentare le targhe delle automobili. Un numero positivo inserisce un'automobile nel vialetto, un numero negativo la fa uscire definitivamente e il numero zero termina la simulazione. Visualizzate il contenuto del vialetto al termine di ciascuna operazione.

★ **E14.11.** Dovete realizzare un "elenco di cose da fare" (*to do list*). A ciascun compito viene assegnata una priorità, un numero intero da 1 a 9, e una descrizione. Quando l'utente digita il comando `add` *priorità descrizione* il programma aggiunge una cosa da fare, mentre quando l'utente digita `next` il programma elimina e visualizza la cosa da fare più urgentemente. Il comando `quit` termina il programma. Risolvete il problema usando una coda prioritaria.

★ **E14.12.** Scrivete un programma che legga un testo da un file e lo suddivida in singole parole. Inserite le parole in un insieme realizzato mediante un albero. Dopo aver letto tutti i dati, visualizzate tutte le parole, seguite dalla dimensione dell'insieme risultante. Questo programma determina, quindi, quante parole diverse sono presenti in un testo.

★ **E14.13.** Leggendo tutte le parole di un file di testo di grandi dimensioni (come il romanzo "War and Peace", *Guerra e pace*, disponibile in Internet), inseritele in due insiemi, uno realizzato mediante tabella hash e uno realizzato mediante albero. Misurate i tempi di esecuzione: quale struttura agisce più velocemente?

★ **E14.14.** Realizzate, nella classe `BankAccount` del Capitolo 8, metodi `hashCode` e `equals` che siano fra loro compatibili. Verificate la correttezza dell'implementazione del metodo `hashCode` visualizzando codici di hash e aggiungendo oggetti `BankAccount` a un insieme realizzato con tabella hash.

★★ **E14.15.** Un punto geometrico dotato di etichetta è caratterizzato dalle coordinate x e y, oltre che dall'etichetta, sotto forma di stringa. Progettate la classe `LabeledPoint` dotata del costruttore `LabeledPoint(int x, int y, String label)` e dei metodi `hashCode` e `equals`: due punti sono considerati uguali quando si trovano nella stessa posizione e hanno la stessa etichetta.

★★ **E14.16.** Realizzate una diversa versione della classe `LabeledPoint` vista nell'esercizio precedente, memorizzando la posizione del punto in un oggetto di tipo `java.awt.Point`. I metodi `hashCode` e `equals` devono invocare gli omonimi metodi della classe `Point`.

★★ **E14.17.** Modificate la classe `LabeledPoint` dell'Esercizio E14.15 in modo che implementi l'interfaccia `Comparable`. Fate in modo che i punti vengano ordinati innanzitutto in base alla loro coordinata x; se due punti hanno la stessa coordinata x, ordinateli in base alla loro coordinata y; se due punti hanno le stesse coordinate, ordinateli in base alla loro etichetta. Scrivete un programma di collaudo che verifichi tutti i casi, inserendo punti in un `TreeSet`.

★ **E14.18.** Aggiungete al valutatore di espressioni visto nel Paragrafo 14.6.3 l'operatore `%`, che calcola il resto della divisione intera.

★★ **E14.19.** Aggiungete al valutatore di espressioni visto nel Paragrafo 14.6.3 l'operatore `^`, che effettua l'elevamento a potenza. Ad esempio, 2 ^ 3 ha come risultato 8. Come in matematica, l'elevamento a potenza deve essere valutato da destra verso sinistra, cioè 2 ^ 3 ^ 2 è uguale a 2 ^ (3 ^ 2) e non a (2 ^ 3) ^ 2 (quest'ultima quantità si può calcolare come 2 ^ (3 × 2)).

★ **E14.20.** Scrivete un programma che verifichi se una sequenza di marcatori HTML è annidata correttamente. Per ogni marcatore di apertura, come `<p>`, ci deve essere un marcatore di chiusura, `</p>`. All'interno di una coppia di marcatori, come `<p> . . . </p>`, possono essere presenti altri marcatori, come in questo esempio:

```
<p> <ul> <li> </li> </ul> <a> </a> </p>
```

I marcatori più interni deve essere racchiusi tra quelli più esterni. Il programma deve elaborare un file contenente marcatori: per semplicità, ipotizzate che i marcatori siano separati da spazi e che al loro interno non ci sia altro testo, ma solo altri marcatori.