Esame Laboratorio di Programmazione

20 Giugno 2014

NOTA *Non è possibile usare le classi del package prog.io del libro di testo*

L'obiettivo è scrivere una applicazione che permette di costruire delle torri usando dei blocchi. Ogni blocco è caratterizzato da un numero e da un colore. I blocchi possono essere di due tipi: blocchi *regolari* e blocchi *jolly (i blocchi jolly sono di colore oro)*. Per costruire le torri, vanno rispettati i seguenti vincoli:

- un blocco regolare può stare sopra a un blocco regolare che abbia lo stesso colore o lo stesso numero, oppure sopra a un blocco jolly;
- un blocco jolly può stare sopra a un qualunque altro blocco.

Un esempio di torre costruita correttamente è (i blocchi sono elencati partendo dalla base della torre fino alla cima):

(42,blu) (92,blu) (92,verde) (11,JOLLY) (19,JOLLY) (61,rosso) (12,rosso) (12,giallo)

Sulla torre è ora possibile porre:
* un blocco regolare di colore giallo o
* un blocco regolare dal numero 12 o
* un blocco jolly.

Le classi da implementare sono le classi descritte sotto. Si noti che:

- Le classi, oltre ai metodi specificati, possono definire altri metodi che si ritengono utili.
- Le operazioni di stampa vanno effettuate esclusivamente nelle classi che definiscono il metodo main.
- Si presuppone l'utilizzo proprio delle parole chiave "public" e "private"…

Classe astratta Blocco

La classe astratta Blocco definisce un blocco. Possiede un solo costruttore che costruisce un blocco di cui vengono specificati il numero (intero positivo) e il colore (una stringa).
La classe definisce il metodo astratto:

- puoStareSopraA(b)

  Metodo astratto che restituisce true se questo blocco può stare sopra al blocco b in base ai vincoli definiti sopra, false altrimenti.

Classe Regolare

Sottoclasse concreta di Blocco che rappresenta un blocco regolare. Possiede un solo costruttore che costruisce un blocco regolare di cui vengono specificati numero e colore.

Classe Jolly

Sottoclasse concreta di Blocco che rappresenta un blocco jolly. I jolly hanno colore oro. Possiede un solo costruttore che costruisce un blocco jolly di cui viene specificato il numero.

Classe Torre

La classe Torre rappresenta una torre. Ogni torre ha un nome che la identifica univocamente (quindi non è possibile inserire due torri con lo stesso nome!). Possiede due costruttori:

- un costruttore che costruisce una torre di cui viene specificato il nome, non contenente alcun blocco;
- un costruttore che costruisce una torre di cui vengono specificati il nome e il blocco alla base (la torre ha quindi un solo blocco).

Non vanno posti limiti sul numero di blocchi che la torre può avere. Oltre ai metodi che si ritengono utili, la classe Torre definisce i metodi:

- addBlocco(b)

  Pone in cima alla torre il blocco b se questo è possibile (ossia, non vengono violati i vincoli di costruzione). Il metodo restituisce true se il blocco è aggiunto, false altrimenti.

Esercizio 1

Scrivere un programma Esercizio1 che costruisce una torre di nome nome_torre. I blocchi utilizzati per la costruzione sono letti da standard input. Ogni linea in input può avere uno dei seguenti formati:

C, nome_torre
+,numero_blocco
+,numero_blocco,colore_blocco
+,numero_blocco
T

Il primo elemento della linea definisce l'operazione da compiere.

Le linee che iniziano con C richiedono la creazione di una torre dal nome nome_torre. *Si noti che i successivi comandi vanno eseguiti sulla torre creata con il comando*

C,nome_torre. Ovviamente i comandi non vengono eseguiti fin tanto che la torre non è stata creata!

Le linee che iniziano con + richiedono l'inserimento di un blocco in cima alla torre. In particolare la linea

+,numero_blocco,colore_blocco richiede di porre in cima alla torre il blocco

regolare avente numero e colore specificati. Le linee

+,numero_blocco

+,numero_blocco, oro

richiedono di aggiungere un blocco jolly avente il numero specificato. Se l'operazione è possibile, il blocco viene inserito in cima alla torre, altrimenti viene stampato il messaggio

Non è possibile aggiungere blocco (numero_blocco,colore_blocco)

La linea T stampa:

1) il nome della torre
2) il numero complessivo di blocchi (regolari e jolly) contenuti nella torre
3) i blocchi della torre nell'ordine in cui i blocchi sono stati inseriti
4) i blocchi della torre in ordine crescente rispetto al numero che caratterizza ogni blocco. Se due blocchi hanno lo stesso numero, vengono stampati in ordine alfabetico rispetto al loro colore.

Le stampe 3-4 non vengono effettuate se la torre non ha ancora blocchi, ma verrà stampata la linea:
"Torre vuota"

Per chiarezza, prima e dopo la stampa di una torre va stampata una linea separatrice composta da 10 caratteri '*'; inoltre, tra ogni passo di stampa della torre, va stampata una linea composta da 10 caratteri '-' (vedere gli esempi).

Si assume che l'input sia inserito correttamente. Non vanno fatte assunzioni sul numero di linee in input. Si noti che l'input va letto da *standard input*, e non da un file.

Esempio

Supponiamo che le linee sullo standard input siano
C,Mio bel vedere
T
+,1,rosso
+,15,rosso
+,40,blu
T
+,1,verde
+,7,oro
+,25,blu

```
+,25,giallo
+,7,giallo
T
```

Il programma Esercizio1 deve stampare:

```
>>java Esercizio1< in1BIS.txt
**********
TORRE Mio bel vedere
----------
BLOCCHI REGOLARI: 0 JOLLY: 0
----------
Torre vuota
**********
Non e' possibile aggiungere blocco (40,blu)
**********
TORRE Mio bel vedere
----------
BLOCCHI REGOLARI: 2 JOLLY: 0
----------
(1,rosso) (15,rosso)
----------
(1,rosso) (15,rosso)
**********
Non e' possibile aggiungere blocco (1,verde)
**********
TORRE Mio bel vedere
----------
BLOCCHI REGOLARI: 5 JOLLY: 1
----------
(1,rosso) (15,rosso) (7,JOLLY) (25,blu) (25,giallo) (7,giallo)
----------
(1,rosso) (7,giallo) (7,JOLLY) (15,rosso) (25,blu) (25,giallo)
**********

>>
```

L’esempio riportato sopra è rediretto dal file in1BIS.txt (ossia, la linea di comando è java Esercizio1 < in1BIS.txt) e produce l’output salavato in out1BIS.txt.

Se l’input è rediretto dal file in1.txt (ossia, la linea di comando è java Esercizio1 < in1.txt), l’output deve essere come nel file out1.txt. Notate che in quella sequenza di comandi il primo blocco non viene inserito, dal momento che non è ancora stata creata la torre.

Esercizio 2

Scrivere un programma Esercizio2 simile a Esercizio1, ma che permette di creare e operare su più torri. Il formato delle linee in input è:

C,nome_torre
+,nome_torre,numero_blocco,colore_blocco
+,nome_torre,numero_blocco
T
B, nome_torre

Il significato della prima linea è come nell'Esercizio 1 in cui Il secondo elemento di ciascuna linea specifica il nome della torre da creare; *attenzione però che il comando va eseguito solo se non è stata già creata una torre uguale, ovvero una torre con lo stesso nome*.

Per le linea successive i comandi hanno lo stesso significato ma la linea contiene innanzi tutto il nome della torre su cui va effettuata l'operazione. *Tale operazione non viene eseguita se la torre non esiste.*

La linea T stampa tutte le torri in ordine decrescente rispetto alla somma dei numeri contenuti nei loro blocchi.

La linea B stampa l'elenco di tutti i blocchi utilizzati nella costruzione delle torri (l'ordine con cui i blocchi sono è lo stesso descritto in Esercizio1 e il numero complessivo di blocchi (regolari e jolly) utilizzati.

Esempio

Se l'input è rediretto dal file in2.txt l'output deve essere come nel file out2.txt.

Istruzioni per la consegna

File da consegnare (solo il codice sorgente, *non* i file .class):

Blocco.java
Regolare.java
Jolly.java
Esercizio1.java
Esercizio2.java
Torre.java

Eseguire l'upload dei file all'indirizzo http://upload.di.unimi.it autenticandosi per la sessione relativa al proprio turno: Turno A (da ME a RE, E. Casiraghi) o Turno B (da RI a Z, C. Cesa Bianchi).

Non consegnare file che non compilano.