Esame di Programmazione 2

20 settembre 2023

- I file con il codice vanno caricati alla pagina `https://upload.mat.unimi.it/` nella sessione dedicata all'esame. Vanno caricati solamente i file `.java`; i file vanno caricati singolarmente (non inseriti in file `.zip`). In caso vengano consegnate più copie dello stesso file, verà corretta l'ultima versione.
- Prima di caricare un file su upload, verificare che sia compilabile; cancellare o includere in commenti le parti che provocano errori in compilazione. **I file che danno errori in compilazione non verranno valutati**. Può essere utile usare opzione `-Xlint` di `javac`, che controlla l'uso corretto dei tipi generici.
- **Non** vanno definiti package.

Domino

Consideriamo una variante del gioco del domino, che consiste nel porre in sequenza delle tessere rispettando alcuni vincoli. Ogni tessera è suddivisa in due parti, ciascuna delle quali contraddistinta da un colore, e ha assegnato un punteggio. Ci sono tessere speciali, chiamate tessere Jolly, in cui le due parti hanno lo stesso colore; tutte le tessere Jolly hanno lo stesso colore e lo stesso punteggio. Nel seguito usiamo la seguente notazione:

- (c_1, c_2, n) indica una tessera in cui il colore sinistro è c_1, il colore destro è c_2, il punteggio è n;
- $(\star\, c, n)$ indica una tessera Jolly in cui il colore è c e il punteggio n.

Le tessere possono essere ruotate, ad esempio, ruotando la tessera $(\texttt{red}, \texttt{blue}, 10)$ si ottiene la tessera $(\texttt{blue}, \texttt{red}, 10)$; le tessere $(\texttt{red}, \texttt{blue}, 10)$ e $(\texttt{blue}, \texttt{red}, 10)$ vanno considerate **uguali**. Le tessere possono essere aggiunte solo a destra di una sequenza già costruita (quindi, dopo l'ultima tessere inserita). Il domino deve rispettare i seguenti **vincoli**:

(V1) la tessere non devono essere duplicate, a meno che non siano Jolly.

(V2) Due tessere adiacenti che non siano Jolly devono avere le parti a contatto dello stesso colore. Le tessere Jolly possono invece essere adiacenti a qualunque altra tessera.

Un esempio di domino che rispetta i vincoli è:

```
(blue, red, 10) (red, red, 20) (red, blue, 20) (*gold, 100) (*gold, 100) (green, red, 30)
```

È ora possibile aggiungere, dopo la tessera $(\texttt{green}, \texttt{red}, 30)$, una tessera Jolly oppure una tessera in cui una delle due parti ha colore `red` e che non sia già contenuta. Ad esempio, la tessera $(\texttt{red}, \texttt{blu}, 10)$ non può essere inserita in quanto il domino contiene già $(\texttt{blue}, \texttt{red}, 10)$. È possibile inserire la tessera $(\texttt{blue}, \texttt{red}, 15)$, ma occorre ruotarla

in modo che le parti a contatto abbiano lo stesso colore (quindi va inserita la tessera (`red`, `blue`, 15)).
È richiesto di definire alcune classi pubbliche per gestire un domino.

- I costruttori e i metodi descritti nella specifica devono essere pubblici, i campi di una classe devono essere privati.
- Le segnature sono a volte scritte in modo incompleto (mancano alcuni tipi).
- È possibile aggiungere a una classe ulteriori metodi (pubblici o privati); si raccommanda di dare ai nuovi metodi dei nomi significativi, che esplicitino il loro significato, eventualmente scrivere commenti chiarificatori.
- Le classi elencate non devono compiere operazioni di stampa (`System.out.println`, ecc.) o di lettura da standard input; le uniche classi che possono farlo sono le classi che implementano una applicazione (classi con metodo `main`).
- Quando si definisce una classe, alcuni dei metodi ereditati dalle superclassi vanno riscritti **(overriding)**.
- È ammesso che i messaggi stampati dal proprio programma abbiano un formato diverso da quelli riportati nel testo, purché le informazioni siano equivalenti e chiaramente identificabili.

Classe `Tessera`

Classe che descrive una tessera. La classe definisce due costruttori:

- un costruttore che costruisce una tessera di cui si specifica il colore sinistro, il colore destro e il punteggio; i colori sono stringhe, il punteggio è un intero.
- un costruttore che costruisce una tessera in cui le due parti hanno lo stesso colore; il costruttore ha per argomenti il colore e il punteggio.

La classe definisce i seguenti metodi pubblici.

- `coloreSinistro()`, `coloreDestro()`, `punti()`.

  Metodi che restituiscono rispettivamente il colore sinistro, il colore destro e il punteggio di questa tessera.
- `puoStareDopoA(Tessera t)`.

  Restituisce `true` se questa tessera, senza essere ruotata, può essere posta dopo la tessera `t` (ossia, a destra di `t`) in base al vincolo (V2), `false` altrimenti.
- `equals(Tessera t)`.

  Restituisce `true` se questa tessera è uguale alla tessera `t`, secondo la definizione scritta all'inizio del testo, `false` altrimenti. Notare che è sufficiente scrivere un'unica istruzione "`return exprBool`", dove `exprBool` è una opportuna espressione booleana.
- `ruota()`.

  Ruota questa tessera.

Classe `TesseraJolly`

Sottoclasse della classe Tessera che definisce una tessera Jolly. Inizialmente il colore delle tessere Jolly è `gold` e il punteggio 100, tali valori possono essere variati durante l'esecuzione del programma. La classe possiede un solo costruttore privo di argomenti e definisce i seguenti metodi statici pubblici:

- `setColore(col)`

  Stabilisce che il colore delle tessere Jolly è `col`.

- `setPunti(n)`

  Stabilisce che il punteggio delle tessere Jolly è `n`.

Classe `Domino`

Classe che gestisce un domino. La sequenza delle tessere nel domino va rappresentata mediante una lista di tipo `List<Tessera>`. Può essere utile definire un riferimento all'ultima tessera (quella cioè dopo cui va inserita una nuova tessera). La classe possiede un solo costruttore che costruisce un domino vuoto. La classe definisce i seguenti metodi pubblici.

- `add(Tessera t)`

  Inserisce in questo domino la tessera `t` se questo è possibile (vedi vincoli (V1) e (V2)); se necessario, la tessera va ruotata.

  Si consiglia di definire il seguente metodo di supporto:

  - `private Tessera tesseraDaAggiungere(Tessera t)`

    Se la tessera `t` non può essere inserita restituisce `null`. Altrimenti restituisce la tessera da inserire, che può essere `t` stessa oppure `t` ruotata.

- `punti()`

  Calcola il punteggio di questo domino, ottenuto sommando i punti delle tessere in esso contenute.

- `tessere()`

  Restituisce una **nuova** lista contenente tutte le tessere in questo domino (in qualunqe ordine). Notare che non va restituito il riferimento alla lista nel domino, ma va creata una nuova lista.

- `colori()`

  Restituisce la lista dei colori che compaiono nelle tessere in questo domino (in qualunque ordine, ma senza duplicazioni).

```
// *Non* cancellare il codice e i commenti gia' scritti
// Completare il codice come richiesto nelle linee con  ###

public class  Esercizio1 {

 public static void main(String[] args)  {
  Domino dom = new Domino();
  dom.add( new Tessera("green", "blue", 10) );
  dom.add( new Tessera("blue", "pink", 10) );
  dom.add( new Tessera("pink", "red", 20) );
  dom.add( new Tessera("red", "red", 30) );
  dom.add( new Tessera("pink", "green", 30) ); // non aggiunta
  dom.add( new Tessera("blue", "red", 30) );   // ruotata
  dom.add( new Tessera("red", "blue", 30) );   // non aggiunta (duplicato)
  dom.add( new Tessera("red", "blue", 10) );   // ruotata
  System.out.println("(1) " + dom );
  dom.add( new TesseraJolly() );
  dom.add( new Tessera("pink", "red", 20) );  // non aggiunta (duplicato)
  dom.add( new Tessera("red", "pink", 20) );  // non aggiunta (duplicato)
  dom.add( new Tessera("pink", "red", 40) );
  dom.add( new TesseraJolly() );
  dom.add( new TesseraJolly() );
  dom.add( new Tessera("pink", "gray", 20) );
  System.out.println("(2) " + dom );
  System.out.println("(2) PUNTI: " + dom.punti() );
  System.out.print("(2) COLORI: ");
  // ###  stampare i colori nelle tessere (in qualunque ordine, ma senza duplicazioni)
  // ###  il colore delle tessere Jolly deve diventare red
  // ###  il punteggio delle tessere Jolly deve diventare 50
  System.out.println("(3) " + dom );
  System.out.println("(3) PUNTI: " + dom.punti() );
  System.out.print("(3) COLORI: ");
  // ###  stampare i colori nelle tessere (in qualunque ordine, ma senza duplicazioni)
  } // end main

} //end class
```

Figura 1: File `Esercizio1.java`.

Esercizio 1

Completare il file `Esercizio1.java` (vedi Figura 1). Il comando `java Esercizio1` deve stampare:

```
(1) (green, blue, 10) (blue, pink, 10) (pink, red, 20) (red, red, 30)
    (red, blue, 30) (blue, red, 10)
(2) (green, blue, 10) (blue, pink, 10) (pink, red, 20) (red, red, 30) (red, blue, 30)
    (blue, red, 10) (*gold, 100) (pink, red, 40) (*gold, 100) (*gold, 100) (pink, gray, 20)
(2) PUNTI: 470
(2) COLORI: green blue pink red gold gray
(3) (green, blue, 10) (blue, pink, 10) (pink, red, 20) (red, red, 30) (red, blue, 30)
    (blue, red, 10) (*red, 50) (pink, red, 40) (*red, 50) (*red, 50) (pink, gray, 20)
(3) PUNTI: 320
(3) COLORI: green blue pink red gray
```

Esercizio 2

Aggiungere nella classe `Domino` il metodo statico

- `sort(listaTessere)`

  Ordina la lista di tessere specificata dal parametro in questo modo:

  (a) prima vanno poste tutte le tessere Jolly;
  (b) successivamente, tutte le altre tessere, in ordine di punteggio crescente; le tessere con uguale punteggio possono stare in un qualunque ordine.

  L'ordine specificato nei punti (a) e (b) va definito come *ordine naturale (natural ordering)* fra tessere.

Definire la applicazione `Esercizio2` che costruisce un domino come in `Esercizio1` e successivamente stampa le tessere contenute nel domino nell'ordine specificato in (a) e (b). Al termine, va nuovamente stampato il domino; controllare che l'operazione di ordinamento delle tessere non abbia alterato la sequenza delle tessere nel domino.

Esercizio 3

Definire la applicazione `Esercizio3` che costruisce un domino come in `Esercizio1`. Al termine di ciascuno dei punti (1), (2) e (3) vanno stampati i colori nelle tessere del domino ordinati come segue: ordine crescente rispetto alla lunghezza del nome del colore, i colori con la stessa lunghezza vanno ordinati in ordine alfabetico inverso.

Va quindi stampato:

```
(1) (green, blue, 10) (blue, pink, 10) (pink, red, 20) (red, red, 30)
    (red, blue, 30) (blue, red, 10)
(1) COLORI ORDINATI: red pink blue green
(2) (green, blue, 10) (blue, pink, 10) (pink, red, 20) (red, red, 30) (red, blue, 30)
    (blue, red, 10) (*gold, 100) (pink, red, 40) (*gold, 100) (*gold, 100) (pink, gray, 20)
...
(2) COLORI ORDINATI: red pink gray gold blue green
(3) (green, blue, 10) (blue, pink, 10) (pink, red, 20) (red, red, 30) (red, blue, 30)
    (blue, red, 10) (*red, 50) (pink, red, 40) (*red, 50) (*red, 50) (pink, gray, 20)
...
(3) COLORI ORDINATI: red pink gray blue green
```