**Esame di Programmazione II, 5 febbraio 2020**

*Si crei un progetto Eclipse e, nella directory dei sorgenti, si crei il package* `it.univr.cards`. *Si copi al suo interno le classi del compito. Non si modifichino le dichiarazioni dei metodi. Si possono definire altri campi, metodi, costruttori e classi, ma devono essere* `private`. *Questo significa che eventuali classi aggiuntive potranno solo essere interne o anonime. La consegna fornita compila. Anche la soluzione che verrà consegnata dovrà compilare, altrimenti non verrà corretta. Alla fine si consegnino soltanto* `Card.java`, `Deck.java`, `Ranking.java` *e* `Main.java`, *separati (cioè non in un file zip).*

**Esercizio 1** [**6 punti**] La classe `Card` rappresenta una carta del gioco del poker, fatta da un valore (enumerazione `Value`, già fatta e da non modificare: 2,3,4,5,6,7,8,9,10,J,Q,K,1: si noti che l'asso ha il valore massimo) e da un seme (enumerazione `Suit`, già fatta e da non modificare: ♠, ♣, ♦, ♥). La si completi dove indicato con `TODO`.

**Esercizio 2** [**3 punti**] L'enumerazione `Ranking` rappresenta il ranking, o punteggio, di una mano di poker, cioè di cinque carte diverse del mazzo da poker. Si completi questa enumerazione in modo da enumerare le possibilità, oltre a `NONE`, nel seguente ordine crescente per ranking:

`THREE_OF_KIND`: in italiano sarebbe *tris*, cioè tre carte di ugual valore e le altre di valore diverso, come per esempio $[10\spadesuit, J\spadesuit, Q\clubsuit, J\heartsuit, J\clubsuit]$

`STRAIGHT`: in italiano sarebbe *scala*, cioè cinque carte non dello stesso seme che si possono mettere in sequenza, come per esempio $[10\spadesuit, 1\spadesuit, K\spadesuit, Q\heartsuit, J\heartsuit]$ (si ricordi che l'asso ha il valore massimo)

`FLUSH`: in italiano sarebbe *colore*, cioè cinque carte dello stesso seme ma non scala, come per esempio $[5\spadesuit, 6\spadesuit, 10\spadesuit, J\spadesuit, K\spadesuit]$

`FULL_HOUSE`: in italiano sarebbe *full*, cioè un tris e una coppia, come per esempio $[1\spadesuit, 3\clubsuit, 3\heartsuit, 1\clubsuit, 3\diamondsuit]$

`FOUR_OF_KIND`: in italiano sarebbe *poker*, cioè quattro carte dello stesso valore, come per esempio $[K\diamondsuit, K\spadesuit, K\clubsuit, 7\diamondsuit, K\heartsuit]$

`STRAIGHT_FLUSH`: in italiano sarebbe *scala reale*, cioè una scala con tutte le carte dello stesso seme, come per esempio $[9\heartsuit, Q\heartsuit, J\heartsuit, 10\heartsuit, K\heartsuit]$

**Esercizio 3** [**13 punti**] La classe `Deck` rappresenta una mano di poker fatta da cinque carte diverse. La si completi dove indicato con `TODO`. In particolare, vanno completati i costruttori e i metodi di test del ranking, come ad esempio `isStraight()`, che determinano se le cinque carte hanno il ranking corrispondente. Per esempio, `isStraight()` determina se le cinque carte formano una scala. Va completato anche il metodo `getRanking()`, che restituisce il ranking della mano, e il metodo `hasRankingFrom()`, che determina se la mano ha almeno un dato ranking minimo indicato.

**Suggerimenti:**

1. non dimenticatevi di lanciare l'eccezione prevista nel secondo costruttore di `Deck` (guardate i commenti prima di esso)

2. la classe `Deck` dovrà in qualche modo contenere le cinque carte. Riflettete su quale struttura dati sia la più semplice per i vostri scopi e su come può convenirvi mantenere le cinque carte al suo interno;

3. leggete bene i commenti del codice riportati sopra i metodi di test, come `isStraight()`. Tali commenti sono la descrizione di una soluzione semplice per implementare i metodi. Può aiutarvi implementare qualche metodo privato di supporto?

**Esercizio 4** [**10 punti**] Modificate la classe `Main` dove indicato con `TODO`, in modo da farle stampare un output simile a quello che trovate in `stampa.txt`:

1. prima genera e stampa 4 full house a caso
2. poi 6 four of a kind a caso
3. poi 8 straight a caso
4. poi 5 straight flush a caso
5. poi 9 three of a kind a caso
6. poi 5 flush a caso
7. poi 10 mani di poker a caso con ranking da flush in su
8. infine crea tre mani `d1`, `d2` e `d3` e verifica che siano tutte e tre dei tris.

Questa classe chiama ripetutamente il suo metodo `print(howMany, when)`, che dovrete completare. Tale metodo stampa `howMany` esempi di `Deck` a caso che soddisfano la condizione `when`. Tale condizione è un'istanza dell'interfaccia di libreria (già esistente) `java.lang.function.Predicate⟨Deck⟩`, che ha un unico metodo `test(Deck)` per sapere se un deck soddisfa o meno il predicato.

**Suggerimento:** Dovrete passare a `print(howMany, when)` vostre implementazioni dell'interfaccia di libreria `java.lang.function.Predicate⟨Deck⟩`. Ci sono tanti modi per farlo. Ricordatevi che quell'interfaccia ha un unico metodo.