Esame di Programmazione II, 4 settembre 2017

Si consideri un'implementazione di un sistema di prenotazione di compagnie aeree che, per semplicità, gestiscono solo una linea. Deve essere possibile prenotare dei biglietti per una data specifica, purché ci siano ancora posti per quella data. I posti dipendono dall'aereo che la compagnia mette a disposizione per quella data: aerei diversi possono avere un numero di posti diverso. Il prezzo del biglietto dipende da quanti posti ci sono ancora disponibili per la data richiesta.

Esercizio 1 [**4 punti**] Si completi la seguente classe che rappresenta una data. Non ci si preoccupi di verificare che le date siano esistenti:

```
public class Date {
  ...
  public Date(int day, int month, int year) { ... }
  @Override public boolean equals(Object other) { ... }
  @Override public int hashCode() { ... }
  @Override public int compareTo(Date other) { ...mette in ordine cronologico... }
  @Override public String toString() { ...ritorna una stringa del tipo dd/mm/yyyy... }
  public boolean isXmas() { ...determina se this e' il 25/12... }
  public boolean isNewYearsEve() { ...determina se this e' il 31/12... }
  public boolean isThanksgiving() { ...determina se this e' il 23/11... }
  public boolean isAllSaintsDay() { ...determina se this e' l'1/11... }
  public boolean isColumbusDay() { ...determina se this e' il 9/10... }
  public boolean isLincolnBirthday() { ...determina se this e' il 12/2... }
}
```

Esercizio 2 [**3 punti**] Si crei la seguente gerarchia di eccezioni controllate:

- `BookingException`, astratta;
- `QuoteException`, astratta e sottoclasse di `BookingException`;
- `PriceChangedException`, concreta, sottoclasse di `BookingException` con messaggio `"flight price has changed"`;
- `IllegalBookingDatesException`, concreta, sottoclasse di `QuoteException` con messaggio `"illegal booking dates"`;
- `FlightSoldOutException`, concreta, sottoclasse di `QuoteException` con messaggio `"flight soldout"`.

Esercizio 3 [**1 punti**] Si completi la seguente classe che rappresenta un modello di aereo, cioè ne indica il nome e il numero di posti:

```
public class Aircraft {
  ...
  public Aircraft(String name, int capacity) { ... }
  public int getCapacity() { ... }
  @Override public String toString() { ...restituisce il nome dell'aereo... }
}
```

Esercizio 4 [**1 punti**] Si definisca un'interfaccia `Fleet` che specifica la flotta messa a disposizione da una compagnia aerea. Dovrà avere un unico metodo `getAircraftFor()` che, fornita una `Date`, restituisce l'`Aircraft` disponibile per quella data.

Esercizio 5 [**15 punti**] Si completi la seguente classe, che gestisce le prenotazioni in un certo periodo temporale con una data flotta. Specifica anche il costo minimo e massimo dei biglietti in quel periodo:

```
public abstract class Bookings {
  ...
  protected Bookings(Date start, Date end, int minimalPrice, int maximalPrice, Fleet fleet) {
    ...lancia una IllegalArgumentException se start viene dopo di end oppure se minimalPrice e' superiore a maximalPrice o negativo
  }

  // ritorna il prezzo del biglietto per la data indicata
  public int getQuoteFor(Date when) throws QuoteException {
    ...lancia una IllegalBookingDatesException se when non e' tra start ed end incluse
    ...lancia una FlightSoldOutException se per la data when non ci sono piu' posti nell'aereo
    ...ritorna il prezzo del biglietto per la data when, distribuendo uniformemente il prezzo:
    ...se l'aereo e' vuoto, il prezzo e' minimalPrice, se rimane l'ultimo biglietto, il prezzo e' maximalPrice
  }
```

```
  // compra un biglietto per la data indicata,purche' costi il prezzo indicato
  public void book(Date when, int price) throws BookingException {
    ...lancia una IllegalBookingDatesException se when non e' tra start ed end incluse
    ...lancia una FlightSoldOutException se per la data when non ci sono piu' posti nell'aereo
    ...lancia una PriceChangedException se price non e' il prezzo per la data indicata
    ...altrimenti prenota un biglietto per la data indicata
  }

  public int book(Date when) throws BookingException {
    ...lancia una IllegalBookingDatesException se when non e' tra start ed end incluse
    ...lancia una FlightSoldOutException se per la data when non ci sono piu' posti nell'aereo
    ...altrimenti prenota un biglietto per la data indicata, a qualsiasi prezzo, e ritorna tale prezzo
  }

  @Override public String toString() {
    ...ritorna una tabella con le date per cui si sono venduti biglietti, indicando quanti ne sono stati venduti per ogni data
    ...e che tipo di aereo e' previsto per quella data. La tabella deve essere in ordine cronologico
  }
}
```

Esercizio 6 [1 punti] La classe `Bookings` è astratta ma non ha metodi astratti. È corretto? Perché?

Esercizio 7 [4 punti] Si scriva un'implementazione `AirBustFleet` di `Fleet`, per la flotta della compagnia AirBust, che usa i seguenti aerei:

- un A380, con 388 posti, per Natale (25 dicembre), Thanksgiving e San Silvestro (31 dicembre);
- un A330, con 277 posti, per il Columbus day;
- un A320, con 200 posti, per tutti gli altri giorni.

Esercizio 8 [2 punti] Si completi la classe `AirBustBookings` che specifica le prenotazioni della compagnia AirBust per un intero anno indicato, dal primo gennaio al 31 dicembre di quell'anno. I prezzi minimi e massimo dei biglietti sono 100 e 500, rispettivamente. La flotta utilizzata è una `AirBustFleet`:

```
public class AirBustBookings extends Bookings {
  public AirBustBookings(int year) { ... }
}
```

Se tutto è corretto, il seguente programma:

```
public class Main {
  public static void main(String[] args) throws BookingException {
    Bookings bookings = new AirBustBookings(2018); Date Xmas = new Date(25, 12, 2018); Date d = new Date(5, 9, 2018);

    // prenotiamo 300 biglietti per Natale, a qualsiasi prezzo
    for (int count = 0; count < 300; count++)
      System.out.println("Booked Xmas flight at " + bookings.book(Xmas) + " dollars");

    try {
      // prenotiamo altri 800 biglietti
      for (int count = 0; count < 800; count++) {
        bookings.book(Xmas); // per Natale, a qualsiasi prezzo
        int quote = bookings.getQuoteFor(d); // per il 5/9/2018, purche' costino meno di 120 dollari
        if (quote < 120)
          bookings.book(d, quote);
      }
    }
    catch (BookingException e) {
      System.out.println(e.getMessage()); System.out.println(bookings);
    }
  }
}
```

stamperà:

```
Booked Xmas flight at 100 dollars
Booked Xmas flight at 101 dollars
Booked Xmas flight at 102 dollars
.............................
.............................
Booked Xmas flight at 408 dollars
Booked Xmas flight at 409 dollars
flight soldout
 5/ 9/2018: 10 [A320]
25/12/2018: 388 [A380]
```

È possibile definire campi, metodi, costruttori e classi aggiuntive, ma solo `private`.