Esame di Programmazione II, 26 febbraio 2013 (2 ore)

Esercizio 1 [6 punti] Si completi la seguente classe `Time` che rappresenta un istante temporale. Due istanti temporali devono essere `equals` se rappresentano lo stesso momento, esattamente. Il metodo `hashCode` deve essere non banale. Il metodo `compareTo` deve esprimere l'ordinamento cronologico fra istanti temporali. Il metodo `toString` deve restituire una stringa del tipo *yy/mm/dd, hh:mm*, ad esempio `14/2/2013, 21:00`.

```
public class Time implements Comparable<Time> {

  public Time(int year, int month, int day, int hour, int minute) { .... }

  @Override
  public String toString() { .... }

  @Override
  public boolean equals(Object other) { .... }

  @Override
  public int hashCode() { .... }

  @Override
  public int compareTo(Time other) { .... }
}
```

Esercizio 2 [6 punti] Si completi la seguente classe, che implementa un evento che comincia a un certo istante temporale e ha un nome che lo descrive:

```
public abstract class Event implements Comparable<Event> {

  protected Event(Time startingTime, String name) { .... }

  public abstract void delete();

  @Override
  public final String toString() { .... }

  @Override
  public final int compareTo(Event other) { .... }

  @Override
  public final boolean equals(Object other) { .... }

  @Override
  public final int hashCode() { .... }
}
```

Il metodo `toString` restituisce una stringa che fornisce il nome dell'evento e il momento di inizio, del tipo `uscire con la fidanzata @ 14/2/2013, 21:00`. Due eventi sono `equals` se hanno stesso nome e stesso momento di inizio. Il metodo `hashCode` deve essere non banale. Il metodo `compareTo` deve ordinare gli eventi per momento di inizio; a parità di momento di inizio, li deve ordinare in ordine alfabetico per nome. **Suggerimento:** `String` implementa `java.lang.Comparable<String>`.

Il metodo `delete` è astratto e dovrà quindi essere implementato nelle sottoclassi concrete di `Event`. Il suo effetto sarà di cancellare l'evento, eliminandolo dal calendario a cui appartiene.

Esercizio 3 [10 punti] Si definisca una classe `Calendar` che implementa una collezione di eventi temporali (un calendario quindi). Deve fornire:

- un costruttore vuoto, che costruisce un calendario inizialmente privo di eventi;
- un metodo `addEvent(Time startingTime, String name)` che prende nota che al momento indicato inizia un evento dal nome indicato;
- un metodo `toString` che restituisce una stringa che enumera gli eventi in calendario, su righe successive.

La classe `Calendar` deve essere iterabile (`java.lang.Iterable`), nel senso che iterando su un calendario si ottengono gli eventi (classe `Event`) in calendario, in ordine cronologico (dal più vecchio al più nuovo). Quindi avrà un altro metodo pubblico oltre a quelli enumerati sopra.

Suggerimento: la classe iterabile di libreria `java.util.TreeSet<C>` realizza un insieme ordinato purché `C` implementi `java.lang.Comparable<C>`. L'ordinamento avviene rispetto al metodo `compareTo` degli elementi dell'insieme. Gli elementi vengono aggiunti con il metodo `add(C elemento)` (se l'elemento c'è già, non fa nulla) e rimossi con il metodo `remove(C elemento)` (se l'elemento non c'è, non fa nulla).

Se tutto è corretto, il seguente programma:

```
import java.util.HashSet;
import java.util.Set;

public class Main {

  public static void main(String[] args) {
    Calendar cal = new Calendar();
    cal.addEvent(new Time(2013, 2, 14, 21, 00), "uscire con la fidanzata");
    cal.addEvent(new Time(2013, 2, 26, 12, 45), "andare in mensa");
    cal.addEvent(new Time(2013, 2, 27, 9, 10), "andare a sciare");
    cal.addEvent(new Time(2013, 2, 26, 10, 00), "esame di Programmazione II");
    cal.addEvent(new Time(2013, 2, 26, 12, 45), "chiamare la mamma");
    cal.addEvent(new Time(2013, 3, 4, 8, 30), "ricominciare ad andare a lezione");

    System.out.println(cal);

    // ok, la fidanzata e' andata a sciare con qualcun altro:
    // collezioniamo tutti gli eventi che riguardano sciare o che hanno
    // a che fare con la ex-fidanzata....
    Set<Event> deleteThese = new HashSet<Event>();
    for (Event e: cal)
      if (e.toString().contains("sciare") || e.toString().contains("fidanzata"))
        deleteThese.add(e);

    // ....e rimuoviamoli!
    for (Event e: deleteThese)
      e.delete();

    System.out.println(cal);
  }
}
```

stamperà:

```
uscire con la fidanzata @ 14/2/2013, 21:00
esame di Programmazione II @ 26/2/2013, 10:00
andare in mensa @ 26/2/2013, 12:45
chiamare la mamma @ 26/2/2013, 12:45
andare a sciare @ 27/2/2013, 9:10
ricominciare ad andare a lezione @ 4/3/2013, 8:30

esame di Programmazione II @ 26/2/2013, 10:00
andare in mensa @ 26/2/2013, 12:45
chiamare la mamma @ 26/2/2013, 12:45
ricominciare ad andare a lezione @ 4/3/2013, 8:30
```