Esame di Programmazione II, 24 giugno 2013 (2 ore)

Le note musicali sono normalmente divise in 12 *semitoni*, come nella seguente tabella:

semitono	nome italiano	nome inglese
0	do	C
1	do#	C#
2	re	D
3	re#	D#
4	mi	E
5	fa	F
6	fa#	F#
7	sol	G
8	sol#	G#
9	la	A
10	la#	A#
11	si	B

Esercizio 1 [**2 punti**] Si completi la seguente classe che rappresenta una nota musicale:

```
public abstract class Nota {
  protected Nota(int semitono) { ... }

  @Override
  public abstract String toString();

  public abstract Nota incrementa(int inc);

  protected final int getSemitono() { ... }
}
```

dove `getSemitono()` restituisce il semitono della nota, che era stato passato al costruttore. Il metodo `incrementa()`, che deve essere implemetato nelle sottoclassi concrete, restituisce una nota ottenuta aumentando il semitono della nota di `inc` semitoni. Si noti che la scala è circolare per cui, dopo il *si*, si ritorna sul *do*.

Esercizio 2 [**4 punti**] Si scrivano le sottoclassi concrete `NotaIT` e `NotaUK` di `Nota` che si differenziano per il metodo `toString()`, che restituisce una rappresentazione della nota usando, rispettivamente, la notazione italiana e quella inglese.

Esercizio 3 [**4 punti**] Si consideri l'interfaccia:

```
public interface Canzone extends Iterable<Nota> {
  public String getNome();
}
```

che implementa una canzone. Il metodo `getNome()` restituisce il nome delle canzone. Iterando su una canzone si deve iterare sulle sue note.

Si implementi quindi una sottoclasse astratta di `Canzone`:

```
public abstract class AbstractCanzone implements Canzone {

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

completando il metodo `toString()` in modo che restituisca il nome della canzone e il nome delle note usate nella canzone, nel loro ordine di iterazione.

Esercizio 4 [**5 punti**] Si completi la seguente classe concreta:

```
public class CanzoneImpl extends AbstractCanzone {
  public CanzoneImpl(String nome, Nota... note) { ... }
```

```
  @Override
  public String getNome() { ... }

  @Override
  public Iterator<Nota> iterator() { ... }
}
```

in modo che al costruttore vengano passati il nome della canzone e la sequenza di note che la compongono. Se si itera su questa canzone, saranno quelle le note che verranno ottenute dall'iterazione.

Esercizio 5 [7 punti] Si completi la seguente classe concreta:

```
public class CanzoneRibasata extends AbstractCanzone {
  public CanzoneRibasata(Canzone base, int inc) { ... }

  @Override
  public Iterator<Nota> iterator() { ... }

  @Override
  public String getNome() { ... }
}
```

che definisce una canzone identica a `base` ma le cui note sono spostate di `inc` semitoni rispetto a quelle di `base`.

Se tutto è corretto, il seguente programma:

```
public class Main {
  public static void main(String[] args) {
    Canzone boccaDiRosa = new CanzoneImpl
      ("Bocca di Rosa", new NotaIT(11), new NotaIT(2), new NotaIT(3), new NotaIT(6));
    Canzone letItBe = new CanzoneImpl("Let it be", new NotaUK(3), new NotaUK(5), new NotaUK(0));

    System.out.println(boccaDiRosa);
    System.out.println(letItBe);
    System.out.println(new CanzoneRibasata(boccaDiRosa, 5));
    System.out.println(new CanzoneRibasata(letItBe, 23));
  }
}
```

stamperà:

```
Bocca di Rosa:
si
re
re#
fa#

Let it be:
D#
F
C

Bocca di Rosa:
mi
sol
sol#
si

Let it be:
D
E
B
```