**Esame di Programmazione II, 24 settembre 2012**

Si consideri la seguente classe, che rappresenta una sequenza di numeri interi, stampabile e su su cui è possibile iterare:

```
public abstract class Numbers implements Iterable<Integer> {

  @Override
  public final String toString() {
    String result = "";

    for (Integer i: this)
      result += i + " ";

    return result;
  }
}
```

Una sua estensione, che rappresenta la sequenza infinita di numeri interi da una costante `start` in su, è:

```
import java.util.Iterator;

// i numeri interi da una costante in poi
public class From extends Numbers {

  private final int start;

  public From(int start) {
    this.start = start;
  }

  @Override
  public Iterator<Integer> iterator() {
    return new Iterator<Integer>() {

      private int next = start;

      @Override
      public boolean hasNext() {
        return true;
      }

      @Override
      public Integer next() {
        return next++;
      }

      @Override
      public void remove() {
        throw new UnsupportedOperationException();
      }

    };
  }

}
```

**Esercizio 1** [**6 punti**] Si implementi una sottoclasse `UpTo` di `Numbers` con un costruttore `UpTo(int max, Numbers base)`. Essa deve rappresentare la sequenza di numeri interi di `base` fino al primo che supera `max`.

**Esercizio 2** [**8 punti**] Si implementi una sottoclasse `Concat` di `Numbers` con un costruttore `Concat(Numbers first, Numbers second)`. Essa deve rappresentare la sequenza di numeri interi `first` seguita dalla sequenza di numeri interi `second`.

**Esercizio 3 [8 punti]** Si implementi una sottoclasse `Alternate` di `Numbers` con un costruttore `Alternate(Numbers first, Numbers second)`. Essa deve rappresentare la sequenza di numeri interi presi alternativamente da `first` e da `second` (uno da `first`, uno da `second`, uno da `first`, uno da `second`, ecc.). Se una delle due sequenza finisce prima dell'altra, si continua con gli elementi della sequenza più lunga.

Se tutto è corretto, l'esecuzione del seguente programma:

```
public class Main {

  public static void main(String[] args) {
    Numbers from8To25 = new UpTo(25, new From(8));
    System.out.println(from8To25);
    Numbers from100To120 = new UpTo(120, new From(100));
    System.out.println(from100To120);
    Numbers concat = new Concat(from8To25, from100To120);
    System.out.println(concat);
    Numbers alternate = new Alternate(from8To25, from100To120);
    System.out.println(alternate);
    System.out.println(new UpTo(100, alternate));
  }

}
```

stamperà:

```
8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120
8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120
8 100 9 101 10 102 11 103 12 104 13 105 14 106 15 107 16 108 17 109 18 110 19 111 20 112 21 113 22 114 23 115 24 116 25 117 118 119 120
8 100 9
```