Esame di Programmazione II, 21 giugno 2016

Un sistema di posta elettronica è formato da tre componenti: il *server*, le *mailbox* e le *email*. In particolare, il server è capace di spedire email e ha memoria di tutte le email che ha spedito. Una mailbox è legata a un server e fornisce una vista di tutte le email ricevute da un dato utente su quel server; inoltre permette di inviare un'email (tramite il server), di rispondere a un'email (reply-to) e di girare un'email a un destinatario (forward). Le email hanno un mittente, dei destinatari, un soggetto e un corpo (testo).

Intendiamo implementare un sistema di posta elettronica, in modo che per esempio sia possibile scrivere un programma del tipo:

```
public class Main {
  public static void main(String[] args) throws UnknownEmailException {
    Server gmail = new Server();
    Mailbox fausto = new Mailbox(gmail, "fausto.spoto@gmail.com");
    Mailbox scarlett = new Mailbox(gmail, "scarlett.johansson@gmail.com");
    Mailbox jennifer = new Mailbox(gmail, "jennifer.lawrence@gmail.com");
    Mailbox tim = new Mailbox(gmail, "tim.cook@gmail.com");
    // fausto -> { scarlett, jennifer }
    Email e1 = fausto.post("new programming language", "I will soon move to Ruby!", scarlett, jennifer);
    // scarlett -> { fausto, jennifer }
    Email e2 = scarlett.replyToAll(e1, "I think Python is much better!");
    // jennifer -> tim
    Email e3 = jennifer.forward(e2, "Hey Tim, look what these people are saying!", tim);
    // tim -> jennifer
    tim.replyToAll(e3, "Thank you Jennifer!\nI will investigate this issue better");
    System.out.println(jennifer);
  }
}
```

Se tutto è corretto, tale programma dovrebbe stampare tutte le email ricevute dalla mailbox di `jennifer`, separate da trattini, in un qualsiasi ordine:

```
Mailbox of jennifer.lawrence@gmail.com:
----------------------------------------
   From: fausto.spoto@gmail.com
     To: scarlett.johansson@gmail.com, jennifer.lawrence@gmail.com
Subject: new programming language

I will soon move to Ruby!
----------------------------------------
   From: scarlett.johansson@gmail.com
     To: jennifer.lawrence@gmail.com, fausto.spoto@gmail.com
Subject: RE: new programming language

I think Python is much better!

>    From: fausto.spoto@gmail.com
>      To: scarlett.johansson@gmail.com, jennifer.lawrence@gmail.com
> Subject: new programming language
>
> I will soon move to Ruby!
----------------------------------------
   From: tim.cook@gmail.com
     To: jennifer.lawrence@gmail.com
Subject: RE: FWD: RE: new programming language

Thank you Jennifer!
I will investigate this issue better

>    From: jennifer.lawrence@gmail.com
>      To: tim.cook@gmail.com
> Subject: FWD: RE: new programming language
>
> Hey Tim, look what these people are saying!
>
> >    From: scarlett.johansson@gmail.com
> >      To: jennifer.lawrence@gmail.com, fausto.spoto@gmail.com
> > Subject: RE: new programming language
> >
> > I think Python is much better!
> >
> > >    From: fausto.spoto@gmail.com
> > >      To: scarlett.johansson@gmail.com, jennifer.lawrence@gmail.com
> > > Subject: new programming language
> > >
> > > I will soon move to Ruby!
```

Le email vengono modellate dalla seguente interfaccia:

```
import java.util.Set;

public interface Email {
  Mailbox getSender();
  Set<Mailbox> getRecipients();
  String getSubject();
  String getBody();
  String toString();
}
```

Esercizio 1 [7 punti] Si completi l'implementazione di un server. Il metodo `post` serve a spedire un'email, cioè ad aggiungerla a quelle già inviate con il server:

```
import java.util.Set;

public class Server {
  ...
  public void post(Email email) { aggiunge l'email a quelle spedite con questo server... }
  public Set<Email> getEmailsTo(Mailbox recipient) { restituisce le email ricevute da recipient... }
}
```

Esercizio 2 [22 punti, oppure 10 punti per chi ha già fatto il progetto degli anni passati e non dovrà implementare i metodi replyToAll() e forward()] Si completi la seguente classe che implementa una mailbox. Una mailbox si costruisce specificando il server a cui è connessa e l'utente che la possiede. Userà tale server per spedire le email. Una mailbox permette di inviare, rispondere e girare email e ha tre metodi a tale scopo. Si noti che il `toString()` di tali email si dovrà comportare come nell'esempio della pagina precedente. In particolare, attenzione al fatto che le risposte e i forward vengono stampati con sotto l'email a cui si è risposto o che si è girata, con dei caratteri di indentazione `>` alla sinistra.

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

public class Mailbox {
  ...
  public Mailbox(Server server, String user) { ... }
  public Set<Email> getEmails() { restituisce le email ricevute da questa mailbox... }
  public Email post(String subject, String body, Mailbox... recipients) {
    invia e ritorna un'email da questa mailbox ai recipients indicati, con soggetto e corpo indicati...
  }
  public Email replyToAll(Email email, String body) throws UnknownEmailException {
    invia e ritorna una risposta a tutti i destinatari dell'email fornita come parametro,
    meno questa mailbox ma incluso il mittente. Il soggetto della risposta
    sara' quello dell'email fornita come paramero, preceduto da RE:
    Se l'email fornita come parametro non e' stata ricevuta da questa mailbox,
    lancia una UnknownEmailException
  }
  public Email forward(Email email, String body, Mailbox recipient) throws UnknownEmailException {
    invia e ritorna un forward dell'email fornita come parametro al recipient indicato. Il soggetto
    del forward sara' quello dell'email fornita come paramero, preceduto da FWD:
    Se l'email fornita come parametro non e' stata ricevuta da questa mailbox,
    lancia una UnknownEmailException
  }
  @Override
  public String toString() {
    restituisce una stringa con le email ricevute da questa mailbox, separate da trattini.
    Nel mezzo, ciascuna email verra' stampata chiamando toString() sull'email
  }
}
```

Esercizio 3 [2 punti] Si implementi l'eccezione controllata `UnknownEmailException`.

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