Exercises Up To: Chapter 11

This exercise sheet is about basic practice with classes and it's based around email. Note you are not actually going to send email - just write classes that *model* email.

1. An email address is a string with a particular format. Minimally, it has a local part (before the @) and a domain (after the @). Write a class that represents an email address. Your class should have a constructor that accepts two strings representing the local part and domain, get methods that retrieve the local part and domain, and a `toString()` method that returns a string of the form `local-part@domain.`
2. **Challenge** - there are rules defining what can be in an email address (look them up). Extend your class so it checks some or all of these, and rejects illegal email addresses (or at least some of them).
3. Write a class defining an email. An email should minimally have:
   - A subject
   - A recipient
   - A body

   Your class should have methods for creating and/or setting the various parts on the email, and also extracting the various parts (i.e 'get' methods). Your email class should use your email address class - that is, *you should **not** write code that means you can do things like this*:

   ```
   Email exampleMail = new Email("bob@somewhere.com", "Hi there!", "This
   is an email");
   ```

   but instead things like this:

   ```
   Email exampleMail =
      new Email(new EmailAddress("bob","somewhere.com", "Hi there!",
   "This is an email");
   ```

4. Modify your `Email` class so instead of a single recipient, it can handle a list of recipients (remember from Sheet 7 what I said about what I meant when I said 'list'!). The way to do this is hierarchically - don't try to modify your `Email` class constructor so it can handle a list of email addresses directly - instead define an EmailAddressList class that does it for you, and then modify your `Email` class so that it works with `EmailAddressList` objects instead of 'simple' `EmailAddress` objects. E.g. change the above to:

   ```
   EmailAddressList recipientList = new EmailAddressList();
   recipientList.addRecipient(new EmailAddress("bob","somewhere.com");
   Email exampleMail = new Email(recipientList, "Hi there!",
      "This is an email");
   ```

   Doing it like this means you can call `addRecipient` multiple times before creating the email. The idea is each more complex 'thing' has it's own class - you don't try to deal with more than one thing at a time in any one class: each new concept/layer of complexity has it's own. HINT: internally, an `EmailAddressList` is basically just an `ArrayList` of `EmailAddress` objects.
5. Extend your `Email` class so it can handle CC, BCC and Reply-To email addresses (the first two should be lists of addresses, but ReplyTo is just one `EmailAddress` object. Since you don't always want to include these things, you should have multiple constructors - one 'basic' one with just recipients, subject and body, one which also has CC and BCC, and one with CC, BCC and ReplyTo
6. Create an `EmailBox` class that can contain multiple emails. It should be possible to add and remove emails from an `EmailBox`, and an `EmailBox` should have a name (so you'll need a method to get the name).
7. Create an `EmailAccount` class. This should contain two `EmailBox` objects - one called `inBox` and one called `sent`. It should be possible to 'send' an email by adding it to the sent

mailbox; to ‘check’ for mail (it can be an empty method for now); and to delete mail from `inBox` (but *not* from sent).

8. **Challenge -** add various other features of emails and email accounts (e.g. you could have a signature, user-created mailboxes, methods to move between mailboxes).