6 Circular Data

We will now see in practice how to deal with circularly referential data similar to what we have seen in lectures.

In this part we'll visit a familiar concept where circular data exists – namely, lists of friends , or *buddy lists*. These buddy lists could be IM buddy lists, ICQ buddy lists, or lists of friends on social networks. Intuitively a buddy list is just a username and a list of other buddies; the latter part is where we get circularity. So, start by working with the following files:

- *Person.java*
- *ILoBuddy.java*
- *MTLoBuddy.java*
- *Examples.java*

A. Create a project *LabBuddies* and import the four files listed above into the default package. Add the *tester.jar* library to the project as you have done before.

   All errors should have disappeared and you should be able to run the project.

B. Before we can design any methods for the lists of buddies, we need to be able to make examples of buddy lists.

   Design the method `add` that adds a buddy to one person's *buddy list*.

C. Add any additional methods you may need to make sure you can represent the following circle of buddies:

   - Ann's buddies are Bob and Cole
   - Bob's buddies are Ann and Ed and Hank
   - Cole's buddy is Dan
   - Dan's buddy is Cole
   - Ed's buddy is Fay
   - Fay's buddies are Ed and Gabi
   - Gabi's buddies are Ed and Fay
   - Hank does not have any buddies

- Jan's buddies are Kim and Len
- Kim's buddies are Jan and Len
- Len's buddies are Jan and Kim

If someone wants to invite a lot of friends to a party, he or she calls all people on his or her list of buddies, and asks them to invite their friends (buddies) as well, and ask their friends to invite any of their friends as well (anyone that can be reached through this network of friends). We call those on the person's buddy list the *direct buddies* and the others that will also be invited to the party the *distant buddies*.

Now we would like to ask some pretty common questions. For each question design the method that will find the answer. The purpose statements and the headers for the methods are already given:

D. Does this person have another person as a direct friend?

```
// returns true if this Person has that as a direct buddy
boolean hasDirectBuddy(Person that)
```

E. How many direct buddies do the two persons have in common?

```
// returns the number of people that are direct buddies
// of both this and that person
int countCommonBuddies(Person that)
```

F. Will the given person be invited to a party organized by someone?

```
// will the given person be invited to a party
// organized by this person?
boolean hasDistantBuddy(Person that)
```

G. (Optional – be careful!) How many people will be at the party if all those a person invites (directly or indirectly) show up?

```
// returns the number of people who will show up at the party
// given by this person
int partyCount()
```

Follow the Design Recipe!