Appendices F

Iterator Pattern

While Loop

This loop will loop over a set of statements as long as the while condition is true, this type of loop is very powerful as it will only loop over the statements if the condition is true, and can be used to create an infinite loop by giving the while condition true.

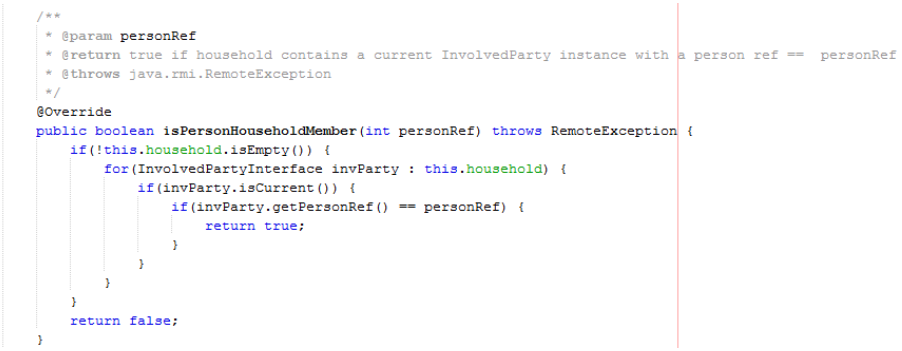
```
/**
 * returns a list of applications with an InvolvedParty that is associated to any Person within the tempPeople list
 * @param tempPeople
 * @return
 * @throws RemoteException
 */
public List<ApplicationInterface> getPeopleApplications(List<PersonInterface> tempPeople) throws RemoteException {
    List<ApplicationInterface> tempApplications = new ArrayList<>();
    if (!tempPeople.isEmpty()) {
        for (ApplicationInterface temp : this.getApplications()) {
            boolean cont = true;
            int i = 0;
            while (cont && i < tempPeople.size()) {
                PersonInterface tempPerson = tempPeople.get(i);
                if (temp.isPersonHouseholdMember(tempPerson.getPersonRef())) {
                    tempApplications.add(temp);
                    cont = false;
                }
                i++;
            }
        }
        return tempApplications;
    }
    return null;
}
```

Fig x – Extract from Database, getPeopleApplications() method

This enhanced for loop from Fig x is within the Database class and as you can see I have used both the enhanced for loop to traverse over a list of all of the system applications, and again I have used this because I don't need to know any information about the position of the element in the list, I just need the element in the list. I then decided to use the while loop because the system should only traverse over the list if the integer called i is smaller than the size of the people list, which ensures that after each iteration through the while block (where i is incremented by 1) there is still another element in the list. However, because I am checking to see if any person from the list of people is within the application as an InvolvedParty, as soon as I come across a person object that is/was a household member on the application, I don't need to continue searching through the list of people so I also need an indicator which is always true unless I have come across a Person element which is within the current application. I used a Boolean field called cont, which is made true if a person is on the application, which on the next check of the while statement will cause the system to not go execute the while statement code and move on to the next application within the list of applications.

Enhanced for loop

This loop is a simpler way of doing the standard for loop and traversing a list, however is not flexible and should only be used when you need to loop over all of the elements within a list, and don't need to know the index of the object you are retrieving [43].



```
/**
 * @param personRef
 * @return true if household contains a current InvolvedParty instance with a person ref ==  personRef
 * @throws java.rmi.RemoteException
 */
@Override
public boolean isPersonHouseholdMember(int personRef) throws RemoteException {
    if(!this.household.isEmpty()) {
        for(InvolvedPartyInterface invParty : this.household) {
            if(invParty.isCurrent()) {
                if(invParty.getPersonRef() == personRef) {
                    return true;
                }
            }
        }
    }
    return false;
}
```

Fig x – Extract from InvolvedPartyImpl, isPersonHouseholdMember() method

This enhanced for loop from Fig x is within the InvolvedPartympl class and I decided to use it here because it is a simpler form of the for each loop and in this instance I don't need to know what the index is of the element within the list as I am just invoking the isCurrent() method on the element to check to see if the involvedParty element is current, and then if yes, invoking the getPersonRef() method on the InvolvedParty element to determine if the personRef provided as a parameter to the method is equal to the personRef return and if so return true. This type of loop has been used a number of times during the project.